13

ENE 06



el aventurero

HISTORIA DE LA AVENTURA

análisis

DON QUIJOTE DE LA MANCHA

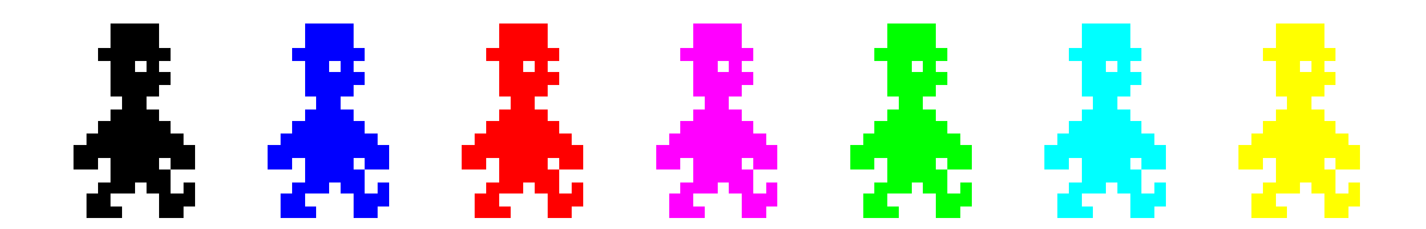
programación z88dk

LA LIBRERÍA SPRITE PACK

programación en ensamblador

ARQUITECTURA Y FUNCIONAMIENTO DEL SPECTRUM

http://magazinezx.speccy.org



Hola. Ya estamos aquí con una nueva entrega de MagazineZX, la 13 para ser exactos (nosotros no somos supersticiosos). Esta vez sin la presión de una fecha concreta, y parece que nos ha ido mejor que la última vez.

Este número está dedicado a las aventuras conversacionales en general y a un clásico de la literatura convertido en videojuego, Don Quijote. Acabamos de celebrar el cuarto centenario de la gran obra de Cervantes y nada mejor que unirnos a la conmemoración con un exhaustivo análisis del juego que Dinamic creó hace 18 años.

Otro gran creador de aventuras conversacionales de antaño, que sigue activo en la actualidad, es Josep Coletas. A él le hacemos una extensa entrevista, como siempre a cargo de Horace, y también hemos querido aprovechar la ocasión para ofreceros la solución completa de dos de las aventuras de su Dr. Van Halen: "Los Cantos de Anubis" y "Tristes Alas del Destino".

Para completar el viaje, Carlos Sánchez nos habla de la historia de la aventura conversacional como género, desde su nacimiento hasta la actualidad, y Juan Pablo López nos relata a su vez la historia del Viejo Archivero, una de las secciones más carismáticas de la revista Microhobby.

Pero la revista no es sólo aventura. También encontraréis los habituales cursos de programación en C y ensamblador. En esta entrega, Pablo Suau profundiza en el uso de la librería Sprite Pack, que nos permitirá un manejo cómodo de los sprites para nuestros juegos. Por su parte, Santiago Romero describe la arquitectura de nuestro Spectrum, ya que su comprensión es la mejor manera de sacarle todo el partido programándolo.

Como siempre, esperamos que os guste el material y nos vemos en el próximo número. ¡Feliz año 2006!

Redacción de MAGAZINE ZX







Año IV nº13 Ene 06

Redacción

Miguel A. García Prada
DEVIL_NET

Pablo Suau
SIEW

Federico Álvarez
FALVAREZ

Ilustración de Portada

Juanje Gómez
DEV

Colaboraciones en este número

Santiago Romero
NOP

Josetxu Malanda
HORACE

Carlos Sánchez
UTO

Juan Pablo López-Grao

Maquetación en PDF

Javier Vispe
ZYLOJ

Contacto
magazinezx@gmail.com

índice

panorama

Como es habitual, desde esta sección hacemos un breve repaso a aquellos acontecimientos más importantes que han tenido lugar desde la anterior publicación de nuestra revista.

DÉCIMO ANIVERSARIO DE WOS

El mes de noviembre del pasado año, la página decana del mundo del Spectrum, WOS (World Of Spectrum) cumplió 10 años de vida, lo cual es mucho tiempo, no solo para una página de Spectrum sino también para una página web en general. Desde aquí nunca nos cansaremos de agradecer el enorme trabajo de Martijn y sus colaboradores y, como siempre, llamar la atención sobre la gran cantidad de programas que quedan todavía por preservar. ¡Larga vida a WOS!

NUEVAS AVENTURAS DE JOSEP COLETAS

Josep no para. Le ha cogido el gustillo a hacer aventuras y no deja de publicar, lo cual es una gran noticia por partida triple: la producción actual para Spectrum sigue viva, encima dicha producción es nacional (lo cual en el ámbito de las aventuras conversacionales es bastante importante ya que el idioma no resulta un impedimento para disfrutarlas a tope) y, además, las aventuras rayan un buen nivel (es decir, que cantidad no está reñida con calidad en este caso).

Los dos últimos juegos lanzados han sido "Código Secreto Lucybel: Cryogenic" y "El Dr. Fanfalen y la Loca Mansión del Terror". El primero de ellos se trata de una aventura futurista, ambientado en el año 2501 en un futuro donde el hombre convive con otras formas de vida artificial y una pacífica raza alienígena, y en el que la supervivencia es el único objetivo. El segundo es una parodia del personaje del propio Coletas, el Dr. Van Halen, protagonista de otras aventuras, algunas de las cuales diseccionamos en este número de MagazineZX.

Ambas aventuras pueden ser descargadas de la página del CAAD.

SPECCY TOUR

Por fin, con un poco de retraso respecto al calendario habitual, comenzó la edición del Speccy Tour correspondiente a 2005. Este año se han inscrito 80 participantes de muy diversas nacionalidades. Eso sí, la representación nacional sigue siendo bastante elevada, como en años anteriores.

La competición concluirá el 22 de enero y, en el momento de escribir estas líneas, la clasificación está siendo liderada por rebufo, con 762 puntos, seguido de Alex Lux (700 puntos), que ha protagonizado una buena remontada los últimos días, y eyp, con 595 puntos. Podéis seguir la clasificación tanto en la página del Speccy Tour como en el portal Speccy.org. Mucha suerte a todos los participantes.

CERRADO EL PLAZO DE ADMISIÓN DEL CONCURSO DE BASIC 2005

El 4 de enero de 2006 se cerraba el plazo de admisión de participantes en el concurso de BASIC organizado por Bytemaniacos. Este concurso, que goza de buena salud, va ya por su tercera edición. Este año se ha organizado en torno a dos categorías, BASIC puro y BASIC libre, habiéndose presentado 6 programas a la primera categoría y 8 a la segunda. Por tanto son 14 nuevas creaciones de Spectrum, la mayoría de gran calidad, de las que podremos disfrutar.

En este momento se encuentra abierto el plazo de votaciones, que concluirá el 9 de enero.

Hay un estupendo +3 como premio al ganador de la categoría BASIC libre, y un lote de MagazineZX impresas (lo cual nos enorgullece) para el ganador de la categoría BASIC puro. Todo el mundo a votar, y en el próximo número os contaremos quiénes han resultado ganadores.

NOVEDADES COMPUTER EMUZONE

El pasado mes de octubre, los chicos de CEZGS lanzaron la versión Spectrum de Columns, una recreación del clásico de SEGA. El acabado del producto es muy profesional y, aunque el juego ya está un poco manido, invita a echarse unas partidas. Altamente recomendable. La buena noticia no es sólo la aparición de este Columns, sino las intenciones que tiene la gente de CEZGS de seguir dando

guerra durante 2006.

Por otro lado, aunque escapa ligeramente al ámbito de esta revista, también queremos comentar el lanzamiento de un *remake* largamente esperado: Sir Fred. Un juego con una buena calidad gráfica y sonora y, al igual que el original, con una dificultad endiablada.

Tanto Columns como el *remake* de Sir Fred los podréis encontrar también en la página de CEZ. Buscadlos allí ya que no tienen pérdida y, por lo visto, a los administradores de CEZ no le gusta que se enlacen directamente sus contenidos.

MADRISX 2006, YA FALTA MENOS

El tiempo pasa volando, y ya estamos a tan solo dos meses de la celebración de una nueva edición de la MadriSX & Retro. La próxima tendrá lugar el 4 de marzo de 2006. Tras el exitazo de la anterior reunión, todos estamos impacientes de que llegue la fecha para poder juntarnos allí y pasar una interesantísima jornada compartiendo lo que más nos gusta, el amor por el Spectrum. Esperamos veros a todos por allí, no olvidéis la fecha, el 4 de marzo.

CSS CRAP GAME COMPETITION 2005

Este año se celebra la décima edición del torneo de juegos "caca" de comp.sys.sinclair. La idea consiste en hacer el peor juego para Spectrum que nunca se haya hecho. En general suelen ser juegos sencillos que casi siempre logran arrancarnos una sonrisa por lo absurdo de su planteamiento o su jugabilidad.

Hasta este momento se contabilizan 37 títulos participantes, y os recordamos que el plazo de presentación de programas concluye el 31 de enero de 2006.

SORTEO NAVIDEÑO DE SPECCY.ORG

En el mes de noviembre, speccy.org abrió un hilo en el que se invitaba a los usuarios registrados del portal a participar en él, escribiendo lo que les apeteciera acerca del Spectrum. La motivación extra era sortear un lote de juegos entre todos los participantes.

El plazo concluyó el 15 de diciembre y el ganador resultó jgutierrez, quien se llevó para su casa joyitas como Chequered Flag, Flight Simulation, Army Moves, Fernando Martín y alguna otra.

La verdad es que sólo se registraron 30 comentarios, lo que no es una cifra para tirar cohetes, y que nos hace pensar sobre las posibles causas: desconocimiento, desidia o es que realmente somos cuatro gatos.

REDACCION DE
MAGAZINE ZX

MadriSX: http://madrisx.cjb.net/

WOS: http://www.worldofspectrum.org/

CAAD: http://caad.mine.nu/

Speccy Tour 2005: http://www.zxspectrum.homeactionreplay.org/tour/tindex.php

Concurso de BASIC 2005: http://www.bytemaniacos.com/html/basic2005.html

Computer EmuZone: http://www.computeremuzone.com/

MSX Power Replay: http://replay.madrisx.org/

The comp.sys.sinclair Crap Game Competition 2005: http://www.lofi-gaming.org.uk/speccy/cgc2005/

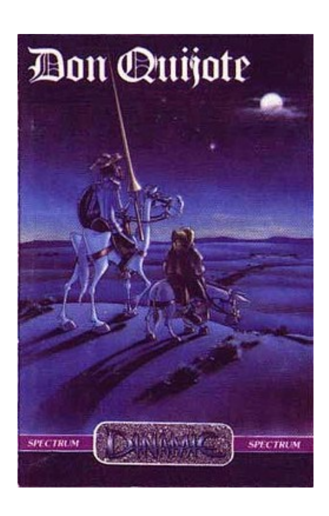
Speccy.org: http://www.speccy.org/





En este número os ofrecemos un prolijo análisis de Don Quijote de la mano de Santiago Romero.

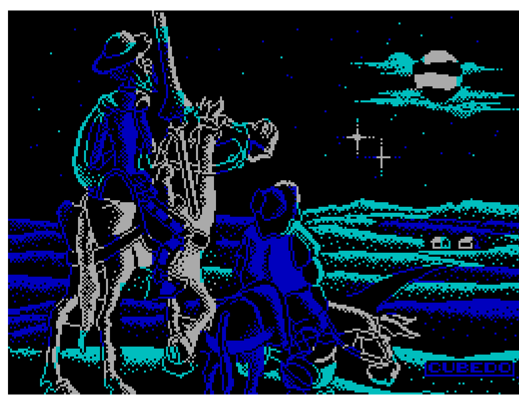
DON QUIJOTE



Título	Don Quijote
Género	Aventura conversacional
Año	1987
Máquina	48K
Jugadores	1 Jugador
Compañía	Dinamic software
Autor	Jorge Blecua (EGROJ), Pablo, Javier Cubedo, Ángel Luis
Otros comentarios	Microhobby 140, página 18 Microhobby 140, página 19

"En un lugar de la Mancha, de cuyo nombre no quiero acordarme, no ha mucho tiempo que vivía un hidalgo de los de lanza en astillero, adarga antigua, rocín flaco y galgo corredor."

Así empieza "El Ingenioso Hidalgo Don Quijote de la Mancha", libro en que está basada la Aventura Conversacional que analizaremos en esta ocasión.



Pantalla de carga de Don Quijote

EL LIBRO

En el año 1605, Miguel de Cervantes y Saavedra publicó la primera edición de su ahora famoso libro "El Ingenioso Hidalgo Don Quijote de la Mancha". Este año 2005 se celebra el cuarto centenario de esta publicación, y desde MagazineZX queremos aprovechar este número dedicado al mundo de las Aventuras Conversacionales para honrar a la vez a la obra de Cervantes y a su equivalente informático en el Spectrum.

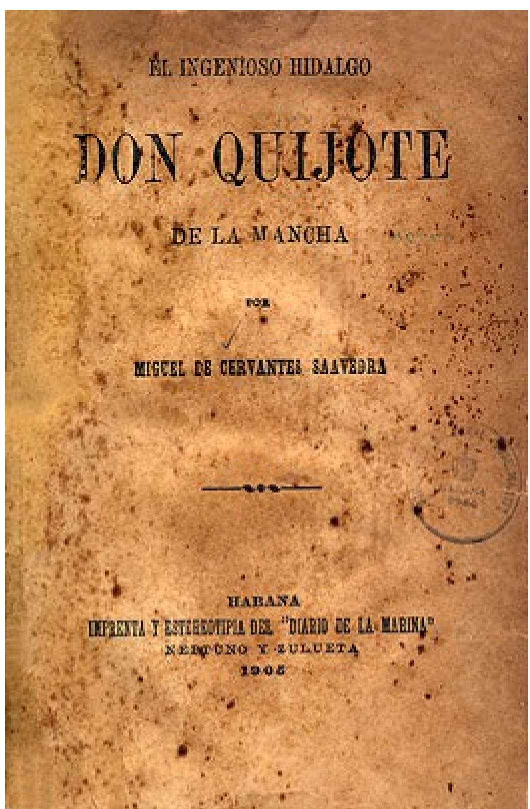
"El Quijote" (el libro) es una de las obras más conocidas de la Historia, traducida a innumerables idiomas y que se erige como el libro más representativo de la Literatura Española, convirtiendo a su autor en, para muchos, el más grande escritor español de todos los tiempos.

La obra es mucho más compleja de lo que aparenta a simple vista, ya que va más allá de ser una simple novela caballeresca. Una buena definición de "El Quijote" la podemos encontrar en la Enciclopedia Universal Micronet:

"El Quijote es la obra maestra de Cervantes y una de las más admirables creaciones del espíritu humano. Es una caricatura perfecta de la literatura caballeresca, y sus dos personajes principales, Don Quijote y Sancho Panza, encarnan los dos tipos del alma española, el idealista y soñador, que olvida las necesidades de la vida material para correr en pos de inaccesibles quimeras, y el positivista y práctico, aunque bastante fatalista.

Considerado en su conjunto, El Quijote ofrece una anécdota bastante sencilla, unitaria y bien trabada: un hidalgo manchego, enloquecido por las lecturas caballerescas, da en creerse caballero andante y

sale tres veces de su aldea en búsqueda de aventuras, siempre auténticos disparates, hasta que regresa a su casa, enferma y recobra el juicio. Sin embargo, el conjunto de la trama no está diseñado de un tirón, sino que responde a un largo proceso creativo, de unos veinte años, un tanto sinuoso y accidentado: cabe la posibilidad de que Cervantes ni siquiera imaginara en los inicios cuál sería el resultado final.



Portada de El Quijote, edición de principios de siglo.

Así pues, "Don Quijote" es una extensa novela caballeresca que narra las aventuras a través de España (principalmente de La Mancha) de Don Quijote y de su fiel escudero Sancho, arrastrados por la locura del primero. Para disfrutar en su justa medida su equivalente informático no os podemos recomendar sino su lectura. La obra está disponible para su lectura en numerosos enlaces en Internet, de forma totalmente gratuita:

- http://www.gutenberg.org/etext/2000
- http://www.el-mundo.es/quijote/

Curiosamente, "El Quijote" fue una de las primeras obras escritas en ser "pirateadas" comercialmente a gran escala (en España), con numerosas ediciones ilegales en imprentas clandestinas. Aunque el precio original de su versión informática en forma de Aventura Conversacional era bastante reducido (875 pesetas de la época), resulta anecdótico que casi 400 años después de su publicación (en 1987) se volviera a repetir la misma situación entre el público español. Muchos de los usuarios de Spectrum (dado el alto índice de piratería de la época) pudieron disfrutar por primera vez de "El Quijote" en formato "original" gracias al número 189 de MicroHobby, en aquel mítico par de cintas que incluía "Don Quijote" y "Supervivencia (El FirFurcio)".

EL JUEGO

En "Don Quijote" asumimos el papel de Don Alonso Quijano, un manchego del siglo XVI enloquecido por la incesante lectura de novelas de caballería. Esta locura es la que lleva a nuestro hidalgo y a su fiel escudero, Sancho, a armarse y correr aventuras a lo largo y ancho de La Mancha.

El juego estaba dividido en 2 partes, cada una de ellas con su correspondiente carga (cara A y cara B de la cinta). En la primera parte del juego, el objetivo consistía en armarse como caballero, consiguiendo todos los elementos necesarios para ello. En la segunda parte (a la cuál se accedía por medio de una clave de texto fácilmente deducible, por cierto) deberemos conquistar el corazón de Dulcinea del Toboso, nuestra amada.



Pantalla de bienvenida a Don Quijote

"Don Quijote" es una aventura conversacional desarrollada por DINAMIC en 1987 basada en el universo de la obra de Cervantes: una adaptación que no sigue las andanzas del libro de forma literal. Fue desarrollada mediante la herramienta G.A.C. (Graphic Adventure Creator), un parser en inglés con el que Jorge Blecua creó, anteriormente a Don Quijote, la aventura Arquímedes XXI. Tras la publicación de "Don Quijote", Blecua formó la compañía Odisea Soft y publicó "Abracadabra", esta vez creada mediante el parser PAWS.

LAS ORDENES

Como en todas las aventuras conversacionales, nos moveremos a través de un entramado de localidades con sus correspondientes descripciones (e imágenes, cuando las haya) mediante los típicos comandos de movimiento ("norte" o "ir norte", "sur", "este", "oeste", "subir", "bajar", "abajo", "arriba") o sus abreviaturas ("n", "s", "e", "ab", "arr", etc.).

Aparte de las acciones típicas, como "mirar" (o "m"), "inventario" (o "i"), "cargar"/"grabar", "acabar" (para abandonar el juego), "examinar" (o "ex"),

"luchar", "abrir", "coger"/"dejar" o "ayuda", entre otras, el G.A.C. (el motor o parser con el que construyó el juego) permite construir frases más o menos complejas separadas con signos de puntuación. El parser ignorará los artículos y adverbios, como en la mayoría de aventuras conversacionales, y ejecutará las órdenes en formato VERBO+NOMBRE que le demos, aunque hayan sido introducidas en una sola sentencia.

Así,la siguiente orden es perféctamente válida:

```
> Examinar estanterias, ir sur, bajar
las escaleras.
```

No se nos pueden olvidar las importantísimas órdenes "comer" y "dormir", ya que en la primera parte de la aventura deberemos alimentarnos regularmente si no queremos morir. La aventura tiene un contador interno de órdenes, de forma que cuando veamos aparecer el mensaje "Empiezas a estar hambriento...", sabemos que nos quedan 10 órdenes o comandos antes del fatídico "No aguantas más sin comer" (al que sigue el "GAME OVER"). En este juego es importante aprovechar al máximo los comandos, en el sentido es que equivocarse de dirección y volver atrás es un error que hay que intentar no cometer. Cualquier comando (excepto las líneas en blanco) aumenta el contador interno y lleva a nuestro Quijote hacia la muerte por inanición. En la segunda parte del juego nos tendremos que preocupar además por el agotamiento (dormir es la solución adecuada en este caso) tras una aparatosa caída.



Los paisajes de La Mancha en la 2a parte de la aventura

Otra orden interesante es "modo", que ejecutada alternará entre modo texto y modo gráficos, es decir, activa o desactiva el dibujado de los gráficos de las localidades, para aquellos que prefieren jugar la aventura en puro modo texto (sólo descripciones).

Más adelante (en las pistas y soluciones) os proporcionamos una lista de comandos que pueden ser algo más complicados de adivinar en determinados momentos del juego.

JUGANDO A "EL QUIJOTE"

La primera parte del juego comienza con una orden especial:

```
> Leer libro
  Empieza la aventura.
```

Tras esto, nos moveremos tanto en los interiores como exteriores de la casa de nuestro hidalgo. El primer objetivo será salir de la misma, para lo cual necesitaremos la llave de la puerta. Pero tratar de coger la llave encierra un gran peligro, del cual deberemos protegernos debidamente. Una vez en el exterior, deberemos armarnos caballero, velando nuestras armas y luchando contra nuestros "enemigos".

La segunda parte del juego nos llevará a recorrer La Mancha con el fin de conseguir una serie de ingredientes con los que fabricar un bálsamo para nuestro hidalgo, teniendo como objetivo conseguir el amor de nuestra amada Dulcinea.

Como ya hemos comentado en el apartado de órdenes, además de estar pendientes de comer y descansar. Esto, unido a las trampas mortales y a la dificultad de acertar con algunos de los comandos de la aventura hace de "Don Quijote" un reto incluso para los que hayan acabado bastantes aventuras conversacionales.



Cerca de la peligrosa alacena

Por último, un detalle curioso de "Don Quijote" es el hecho de que a pesar de representar a uno de los libros más famosos de la Literatura Española, el juego tiene algunos errores ortográficos, como las "obejas", o el mensaje de "Esquisito" (que aparece cuando comemos algo).

AYUDAS Y SOLUCIONES

Aparte de que Sancho nos podrá dar alguna que otra pista con el comando "ayuda", nosotros os vamos a dar una relación de órdenes que deberéis utilizar a lo largo de la aventura; en ocasiones es complicado dar con ellas (la sintaxis exacta). El resto de la aventura, así como mantenerse vivo comiendo y durmiendo, es ya cosa vuestra.

Parte 1:

```
EXAMINAR ESCALON
COGER TABLON
PONER ARMADURA
DEJAR TABLO
GOLPEAR ARBOL
ESCALAR MURO
LLAMAR PUERTA
FORZAR CERRADURA
DEJAR TODO MENOS ESPADA, VELA Y
 CAMISA
VELAR ARMAS
LUCHAR CONTRA ODRES
LUCHAR CONTRA OBEJAS
```
(sí, con B)

Parte 2:

```
MOVER ARBOL
ENTRAR CON CUIDADO EN MOLINO
CANTAR
SILBAR
HACER CAMA
DEJAR (ALGO) EN CALDERO
REMOVER CON RAMITA BALSAMO
```

Tenéis consejos generales y respuestas a preguntas frecuentes (que no solución completa) sobre el juego en las siguientes páginas de MicroHobby:

- Microhobby 140, página 19
- Microhobby 189, página 49
- Microhobby 194, página 47
- Microhobby 198, página 23
- Microhobby 200, página 58

La solución completa la tenéis disponible en la revista Microhobby, números 148 y 149 (sección TOKES & POKES):

- Microhobby 148, página 28
- Microhobby 149, página 28

En cualquier caso, la clave para acceder a la segunda parte de la aventura es:

```
> EL INGENIOSO HIDALGO
```

Con toda esta información, esperamos que todos aquellos que no habéis completado esta difícil aventura conversacional podáis hacerlo y así disfrutar de una de las grandes aventuras de DINAMIC en el cuarto centenario de El Quijote.

Un poco de justificación para las puntuaciones:

Aunque el género en sí (aventura conversacional) no es excesivamente original para la época, el hecho de que esté basado en un libro tan importante y los originales objetivos de cada una de las 2 partes le hacen merecedor de un 8.

Los gráficos están formados, como en la mayoría de las aventuras, de formas geométricas (líneas, rectángulos, etc.) rellenadas de colores planos o de patrones de rellenado. Su dibujado es rápido y están bastante bien conseguidos, lo que le supone otro 8.



El pueblo de nuestro hidalgo.

En cuanto a la jugabilidad y adicción, es un título que invita a ser terminado, especialmente si nos hemos leído el libro. Los dos ochos obtenidos en este apartado podrían haber sido algo más si no fuera por el tema de la dificultad. Las trampas mortales, unidas al hambre y al cansancio, además de algunos comandos complicados de acertar, lo convierten en un juego difícil para todos los que no estén especialmente puestos en las aventuras conversacionales, por lo que le hemos puesto un 9 de dificultad.

Del sonido hay poco que decir, como en la mayoría de conversacionales.

descárgalo de:

WOS

LINKS

http://spa2.speccy.org/spanish/adventure.htm

http://www.arrakis.es/~caad/ficheros/spectrum.html

Valoraciones

originalidad	[8]	jugabilidad	[8]
gráficos	[8]	adicción	[8]
sonido	[0]	dificultad	[9]

SROMERO (NoP)

al descubierto

En este número especial dedicado a las aventuras conversacionales, Miguel G. Prada destripa dos de los extraordinarios casos del Dr. Van Halen, el personaje creado por Josep Coletas: Los Cantos de Anubis y Tristes Alas del Destino.

LOS EXTRAORDINARIOS CASOS DEL DR. VAN HALEN: LOS CANTOS DE ANUBIS

Título	Los Extraordinarios Casos del Dr. Van Halen: Los Cantos de Anubis
Género	Aventura Conversacional
Año	2005
Máquina	ZX Spectrum 48K
Jugadores	1 Jugador
Compañía	J.C.C.
Autor	Josep Coletas Caubet

No puedo quitarme de la cabeza el terrible destino de Hellen. Sueño con ella, las pesadillas me acechan desde el mismo momento en que consigo conciliar el sueño.

Hoy no podía ser una excepción. Me he quedado dormido en mi estudio, con la cabeza reposando sobre los libros. Los vuelvo a colocar en su sitio y me encamino al dormitorio para continuar con mis sueños.

Amanece. No sé si habrá sido el cansancio acumulado o alguna extraña razón, pero he conseguido dormir tranquilo. Me encamino a la planta inferior y el tintineo de la campanilla de la puerta rompe el silencio. Abro la puerta y Laar, el empleado de correos, me entrega un telegrama. Lord Milton Moltimore me cita en su residencia de Londres. El texto de la misiva despierta mi curiosidad.

Me encamino al estudio para recoger mi equipo. En la caja fuerte escondida detrás del retrato de Hellen tengo el dinero. Cojo la linterna y muevo la figura del ángel y el demonio, con lo que se abre el pasadizo hacia la biblioteca secreta. Enciendo la linterna y entro por la chimenea hacia el pasaje a lo desconocido.



Bajo por la escalera de caracol y en la biblioteca sin nombre cojo mi maletín que contiene el catalejo, la moneda Iuramentum y el libro Tenebrarum, herramientas imprescindibles para mi tarea.

Regreso al vestíbulo y cojo del perchero mi bastón, la capa y el sombrero, los cuales me pongo. Antes de abandonar mi hogar echo un vistazo al espejo del tiempo, donde extrañas visiones de ratas y sangre se entremezclan con la realidad. Salgo a la calle y emprendo el viaje hacia Inglaterra.
En el puerto de Londres un viejo marinero toca la armónica y sus sonidos se entremezclan con los de un barco a vapor que se pierde entre jirones de húmeda niebla. Un carruaje me espera para facilitar mis desplazamientos por la ciudad. Subo a él y ordeno al cochero dirigirse a la mansión de Lord Moltimore, en el treinta y tres de Keppel street.



Una verja abierta rodea la mansión. Me dirijo al porche de la casa y toco el timbre de la puerta. Un fuerte ruido de cristales rotos suena en la parte trasera de la mansión; al mismo tiempo unos ladridos de perro se escuchan y la sombra de una figura se adivina saltando las rejas.
Corro hacia la fachada posterior de la mansión y encuentro los cuerpos de dos perros que yacen muertos. El cristal de una ventana de la planta superior está roto. Examino la verja y encuentro un barrote suelto, lo guardo, puede que sea útil más adelante. Por el hueco que ha dejado la verja me introduzco y llego a Torrington square.

Una vez en Torrington square me introduzco por la boca de la alcantarilla. La oscuridad y un intenso olor a excrementos me invade. Enciendo la linterna y examino el techo. Las visiones del espejo del tiempo inundan mi mente... ratas... sangre...

Unas huellas marcadas con sangre me indican la dirección a seguir y me llevan por un ramal del alcantarillado que tiene salida en la parte superior, subo y me encuentro en Charlotte street.

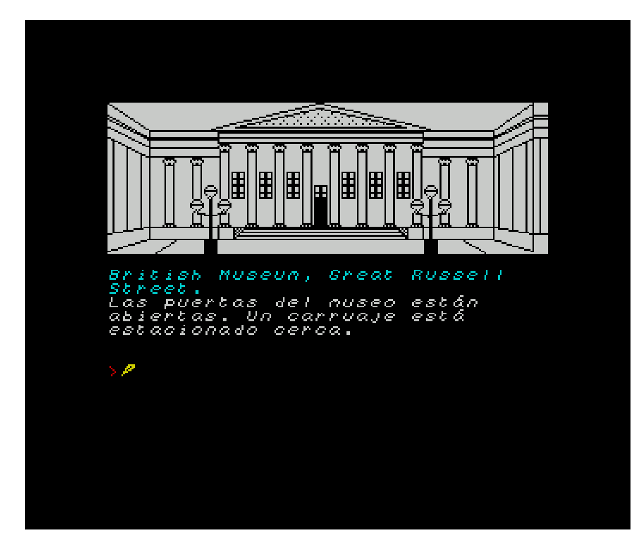
Siento un zarpazo, mi sombrero sale despedido de mi cabeza y un intenso dolor me invade, pierdo el conocimiento. Entre delirios y sueños me despierto en el hospital donde el doctor me informa de que me puedo ir, pero conservando el vendaje que me han puesto.



Cojo todas mis pertenencias, a excepción del sombrero, que perdí cuando me agredieron. Leo el periódico y descubro con horror que Lord Milton ha sido asesinado. Me pongo la capa y el monóculo y salgo de la habitación.

Un carruaje me espera. Le indico la dirección del hotel y me lleva hasta allí. Entro en la habitación y cojo el botellín de whisky. Regreso al carruaje y ordeno que me lleven al puerto.

En el puerto hablo con el marinero, le doy la botella de whisky, vuelvo a interrogarle sin obtener respuesta y subo al carruaje, una voz me saluda. Es Sherlock Holmes, oculto detrás del disfraz de marinero, quien me habla y cita en el Elegant Hat Club.



Otra vez de paseo con el carruaje, en esta ocasión con dirección a la mansión de Lord Milton. Una vez en la entrada un agente me prohíbe el paso, pero tengo que entrar a toda costa. Tomo rumbo a Torrington square y por el hueco que quedó en la verja me introduzco y llego a la parte trasera de la mansión.

Subo por la hiedra con cuidado de no caerme y no alertar a los agentes en la entrada principal, y llego al estudio de la mansión.

Una fastuosa habitación, adornada con numerosos

trofeos de correrías por África. Y el trofeo de un despiadado cazador en el centro de la habitación: el cadáver de Lord Milton.

Hay que ser rápidos. Examino el cadáver y la alfombra, la cual aprecio que han desplazado de su sitio original. Muevo el cadáver y la alfombra y encuentro un trozo de abrigo con el bolsillo intacto. Guardo unas llaves que se encuentran en él y registro el cadáver minuciosamente. El examen da sus frutos y encuentro un ticket perteneciente a la tintorería Coggin's. Bajo por la hiedra y salgo del recinto de la mansión. Desde Torrington street regreso a la entrada de la mansión y subo al carruaje que me espera.

La curiosidad por saber a qué pertenece el ticket me hace dirigirme a la tintorería. Un breve viaje escuchando el traquetear de las ruedas y el casco de los caballos golpeando el empedrado me lleva a Shaftesbury avenue, la dirección de la tintorería. Entro y saludo a una joven que hay detrás del mostrador, me solicita el ticket el cual doy y a cambio recibo un traje de etiqueta. Lo examino pero no encuentro nada de interés en él.

De vuelta al carruaje me dirijo al British Museum, hay una interesante exposición de arte egipcio antiguo. Un impresionante cofre bellamente adornado preside la sala. Está abierto. Entablo una conversación con Sir Shapland sobre la maldición del cofre y nuestro amigo común Lord Milton y se marcha a hablar con un visitante. Examino el cofre y veo un jeroglífico que no me cuesta descifrar, solo hay que leerlo al revés: "Anubis castigará a los no merecedores de su piedad". Enigmático. Espero un poco y el vigilante se marcha a una sala contigua. Abro un sarcófago, me introduzco en su interior y lo cierro. Al poco tiempo de esperar escucho pasos y una conversación que mantienen el vigilante con el director del museo: le da a aquél la noche libre.



Una vez que el vigilante se marcha, salgo del sarcófago y del museo. Las cuatro suenan en el reloj del Big Ben y el British cierra sus puertas al público.
Tenemos que recuperar el sombrero. Voy hacia el oeste y llego a Charlotte street, el lugar en el que me agredieron. Abro la tapa de alcantarilla con el barrote y bajo. Con la luz de la linterna alumbrando las bóvedas del alcantarillado encuentro mi sombrero, el cual cojo y me pongo.

Salgo de la alcantarilla, me visto con el elegante traje de etiqueta y me subo al carruaje rumbo a mi entrevista con Holmes en el Elegant Hat Club.

Una vez en la entrada del club hablo con el portero el cual me deja pasar gentilmente. Sherlock lee el Times acomodado en un sillón y hacia él voy. Me ha citado con Alfred Ganderton en la opera. Me facilita una entrada y me comenta el miedo de Alfred a frecuentar lugares solitarios. Es hora de disfrutar de una velada musical.

Mi obediente cochero me lleva hasta The Royal Opera House en Covent Garden. Entrego la entrada al portero y franqueo la entrada. Fausto vende su alma. Alguien entra en mi palco, le pregunto si es Mr. Ganderton y me dice que no, su nombre es Turner Berriman, abogado. Se lamenta ya que el palco está lejos del escenario y no lo ve bien.

Abro el maletín y cojo el catalejo del interior, se lo doy a Turner. Después de mantener un cruce de palabras con él le pregunto si conoce a Mr. Ganderton. Me dice que sí, que él le dio la entrada para asistir a la ópera en el Pipe Cesar Club. Ya tengo destino.



En el pipe, como indica su nombre, necesito entrar fumando en pipa. Necesito una. Me encamino al Elegant Hat Club y en el lugar donde me entrevisté con Holmes se encuentra su pipa. La recojo y me dirijo a Waterloo, al lugar donde se encuentra el Pipe Cigar Club. Entro. Una densa nube de humo, cual neblina londinense hace que la habitación parezca un pequeño Londres encerrado en una sala. Un hombre fuma nervioso. Hablo con él, es Alfred Ganderton. Está muy asustado, me cuenta una increíble historia sobre el cetro de Anubis y Sir Shapland. Me hace entrega de la copia de las llaves del museo, espero que de la media noche, y con ellas me dirijo allí.

Abro la puerta del museo con las llaves y entro. Me encuentro a Lord Shapland convertido en el enviado de Anubis. Mantenemos una conversación en egipcio:

- ¡Deténgase, Dr. Shapland.
- Su curiosidad ha ido demasiado lejos, Dr. Van Halen.
- ¡Levantaos mis fieles! Grita el enviado.

Las momias se acercan a mi peligrosamente. Tiro de la alfombra del suelo con fuerza y el enviado cae estrepitosamente. El cetro salta de sus manos y llega a las mías, lo golpeo contra el suelo y el enviado de Anubis desaparece envuelto en llamas.

El enfrentamiento con las fuerzas sobrenaturales ha terminado. He vuelto a someterlas. El Dr. Shapland llora con rabia, sus ansias de ser un Dios han quedado disueltas junto a sus lágrimas. Los agentes de Scotland Yard entran por la puerta y se llevan al doctor detenido.

Mi amigo Sherlock hace acto de presencia, como supervisando todos los hechos. Guardo el cetro en el cofre y me lo llevo conmigo. Necesito un descanso, los últimos días han sido muy duros. Voy al hotel en mi carruaje y duermo toda la noche tranquilamente en mi habitación.

El día siguiente es día de viaje. Voy al puerto, mi barco sale con destino al hogar. Entro en mi casa y al llegar al estudio una voz tenebrosa susurra: "... espero que le gustasen los planos de la mansión, Dr. Van Halen."

Descarga el mapa de la aventura:
mapa_anubis.png

LOS EXTRAORDINARIOS CASOS DEL DR. VAN HALEN: TRISTES ALAS DEL DESTINO

Título	Los Extraordinarios Casos del Dr. Van Halen: Tristes Alas del Destino
Género	Aventura Conversacional
Año	2005
Máquina	ZX Spectrum 48K
Jugadores	1 Jugador
Compañía	J.C.C.
Autor	Josep Coletas Caubet

ACTO I: REVELACIONES

Ya no sé si es sueño o realidad, si mi sensatez se torna en locura, si las voces que escucho son reales o si mi razón me abandona poco a poco, al igual que me dejó Hellen, solo, sin rumbo en mi vida... Los planos de la mansión, una voz sacude mis sentidos.

Un día lluvioso en Brujas, típico de esta estación del año. Me dirijo a mi casa de Vlaming straat, abro la puerta y entro. La cierro para que el frío y la lluvia no inunden el vestíbulo y me encamino a mi estudio.

Un relámpago, producto de la tormenta que sacude el exterior, ilumina una espectral figura. Me conmina a seguirla a través de la estantería. Cojo la linterna y empujo la figura del ángel y el demonio que abre el camino secreto hacia la biblioteca. Entro por la abertura que ha dejado la chimenea y enciendo la linterna, que deshace en jirones la oscuridad a mi alrededor. Bajo por la escalera de caracol para entrar en la biblioteca sin nombre.



Cientos de libros recorren las paredes, sus lomos parecen un enorme teclado de piano que no tiene fin. El cofre de Anubis refresca mis recuerdos sobre el anterior caso que resolví, no deberá abrirse jamás. El fantasma me espera y se dirige a mi con una voz de ultratumba y una expresión que no presagia nada bueno.

Después de escuchar la tenebrosa historia por boca del espectro empiezo a entender un poco más la locura que me rodea, la razón de la existencia del espejo del tiempo, la pérdida de Hellen...

Una nueva misión se abre ante mi, el fantasma me urge a cumplir un cometido en Venecia, donde habita su hermana Rosalind y me da esperanzas para recuperar a mi amada. Esto último por si solo me daría fuerzas para cruzar un océano tras otro a nado.



Cojo mi maletín y vuelvo al estudio donde, en la caja fuerte tras el cuadro, está el dinero guardado. Lo cojo ya que necesitaré divisas para los gastos en el viaje. Aunque no es necesario para continuar, la tentación me vence y subo a echar un vistazo al espejo del tiempo. Una bella ciudad a orillas del mar aparece ¿será Venecia, mi destino? Retorno al vestíbulo y salgo a la calle para viajar rumbo a la bella ciudad italiana, viva muestra del renacentismo. En ella pasan los gélidos días previos a la entrada de la primavera, mientras las noches se suceden paseando por sus canales y admirando el Adriático y a la gente con sus disfraces de carnaval. No se nada de Hellen, no se nada del diablo...

ACTO II: LA NOCHE DEL VAMPIRO.

Desde la ventana de mi habitación admiro la luna que ilumina los bellos ríos de agua. Salgo de mi dormitorio y recorro el largo pasillo hasta la escalera que baja hacia la recepción. En ella el dueño de la pensión dormita sobre el mostrador, es evidente que no tiene la misma resistencia al sueño que yo... o es poseedor de una conciencia más tranquila.

Salgo de la pensión y bajo la luz de la luna me dirijo hacia el este, cruzando el puente dell' Abbazia y luego hacia el sur, llegando al puente di Rialto. En él encuentro una soga, la guardo ya que me puede ser útil más adelante, y veo a una pareja que se hospeda conmigo en la pensión mantener una conversación entre enamorados.

Continúo hacia el oeste y luego al sur y llego a la iglesia gótica de los Frari. Una vieja mujer ora en sepulcral silencio que oso interrumpir. "Tráigame una posesión de ella y le descubriré esas sendas, extranjero". Cojo un crucifijo del altar y observo un atril vacío. La puerta de la sacristía está cerrada con llave, quizá más adelante pueda abrirla.

Salgo en dirección sur de nuevo y tras pasar el palazzo Balbi y girar al este llego a la Gallerie dell' Accademia. Estoy echando de menos un carruaje que me traslade por la ciudad, pero en Venecia, con los canales, sería más apropiado una góndola. Tendré que seguir caminando.



Unas estatuas engalanadas con ocasión de los carnavales flanquean la entrada de la Accademia. Una de ellas lleva un antifaz puesto que pasa a ser de mi propiedad, me lo pongo. Al intentar entrar el portero me pide la entrada, por el momento no la poseo. Dirijo mis pasos al norte y tras callejear un poco y girar a la derecha llego a la plaza de San Marcos.

La plaza está repleta de personas celebrando el carnaval con un animado baile. El anonimato que proporcionan las máscaras hacen que la gente pierda la vergüenza. Una misteriosa dama me pide un baile, menciona a Hellen...

Termina el baile y la mujer se dirige al palacio Ducal, abre la puerta y entra. La sigo. El esplendor del Palazzo Ducale me fascina. Mi enigmática dama ha desaparecido de mi vista. Camino al este del palacio, cruzo el puente de los suspiros y llego a la prisión, repleta de mazmorras tenebrosas. Al fondo, entre las luces y sombras retorcidas que crean una multitud de velas alineadas junto a las paredes, veo la silueta de mi amada Hellen. Tanto tiempo esperando este momento...

Hablo con ella, pero rápidamente descubro el engaño. Unas alas se despliegan desde su espalda y su verdadera identidad sale a la luz: Una vez se llamó Rosalind, ahora es Luzibel. Tengo que terminar con ella y liberarla de esta pesadilla.

Mostrando el crucifijo gano algo de tiempo, pulso el botón de mi bastón y la cuchilla emite un sonido

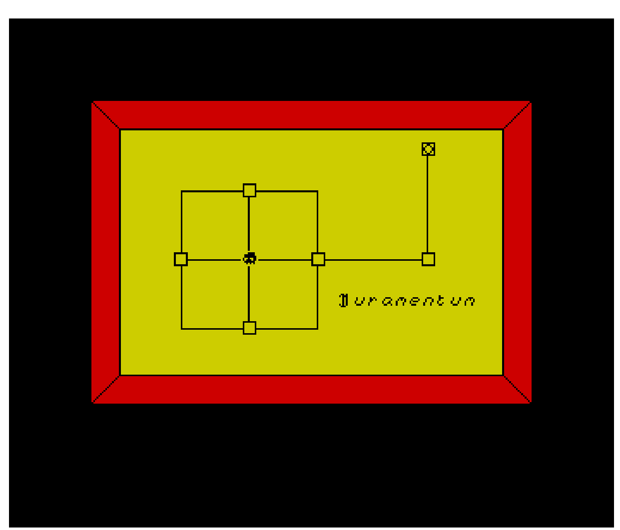
mecánico al salir. Se lo lanzo a la vampiresa y la cuchilla atraviesa su corazón. Los restos de su cuerpo, convertidos en cenizas, caen a la par que escucho un tintineo metálico de algo que golpea el frío suelo. Cojo el bastón y la cadena, la miro atentamente y descubro horrorizado que perteneció a Hellen.



La noche ha sido muy dura, pero creo que esto no ha hecho más que empezar...

Regreso a la iglesia, quiero, necesito saber algo de Hellen por boca de la anciana. Le entrego la cadena y agarrándola con sus ajadas manos la aprieta y cerrando sus ojos me dice: “Busque la Flor del Ocaso, la dama de la noche desvelará su secreto”.

Cada vez que creo ver la luz, un enigma se va abriendo paso tras otro y vuelve a oscurecer mi futuro.



Regreso a mi habitación en la pensión, abro la ventana y salgo por ella a la cornisa. Me desplazo con cuidado de no caer al vacío y me cuelo a la habitación colindante. Encuentro una invitación. Vuelvo a salir por la ventana y regreso a mi habitación por la cornisa. Entro y cierro la ventana. Guío mis pasos de nuevo en dirección a la Gallerie dell’Accademia y le doy la invitación al portero, quien me deja el paso franco para entrar. Una vez dentro una bella colección de arte pictórico se extiende ante mis ojos. Bellos cuadros de arte veneciano de artistas que vivieron entre los siglos XIV y XVIII.

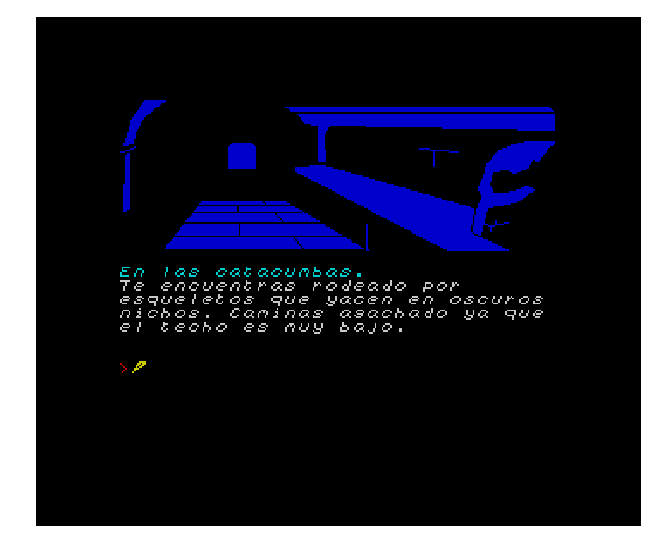
Pregunto al guía, vestido con atuendo religioso, por la “dama de la noche”. Con una mirada sombría me dice que le siga y abre una puerta. Entro, es el almacén. Está repleto de cuadros tapados por lonas, restos de colección guardados para mejores ocasiones. El guía me relata una leyenda en la que me veo reflejado, como si alguien hubiera escrito una novela de la que soy protagonista. Cojo el cuadro y entre frases de buenos deseos del guía me encamino a la iglesia de nuevo.

Una vez en el sagrado edificio coloco el cuadro en el atril vacío. A la luz de la luna el cuadro se transforma, dejando ver unos trazos antiguos.

ACTO III: LA BÚSQUEDA

El cuadro oculta un mapa bajo la imagen de la dama. En él, además de un plano, hay escrita la palabra “Iuramentum”, el nombre de mi moneda, una pista inequívoca sobre lo que tengo que hacer.

Me encamino hacia el campo dell’Abbazia. Recuerdo haber visto un poco en uno de mis paseos por la ciudad. Ato la soga a la argolla y desciendo por la cuerda con sumo cuidado. La oscuridad invade el lugar y noto el agua fría mojando mis piernas hasta las rodillas. Enciendo la linterna y me encuentro en una laguna subterránea. No tengo más alternativas que volver sobre mis pasos trepando por la cuerda o dirigirme a lo que supongo es el sur, a visitar territorio inexplorado. Camino lentamente, iluminado por la luz de mi linterna y deseando que no se agote su combustible.



Después de un tiempo caminando por grutas el camino gira bruscamente hacia el oeste. Tengo la sensación de que me guían como un cordero al

matadero. Estoy en las grutas debajo de la ciudad de Venecia. El silencio se quiebra con el sonido de gotas de agua golpeando el suelo. Sombras en movimiento se escabullen por doquier. El intenso frío hiela mis huesos.

Sigo rumbo oeste y me introduzco de lleno en las antiguas catacumbas, un inmenso laberinto. Siento como si las cuencas vacías de las calaveras se clavaran en mí. Cientos de años me observan desde las paredes. Examino el techo y aprecio una losa. La empujo y un hueco se abre. Me introduzco por él y aparezco en la sacristía de la iglesia. Encuentro un cáliz y una llave, que me guardo. Utilizo la llave para abrir la puerta y salir. Lleno el cáliz con agua bendita, regreso a la sacristía y por el agujero en el suelo regreso a las catacumbas.

Mi instinto me hace dirigirme al sur, lanzo la moneda Iuramentum y desaparece con un parpadeo. En su lugar se encuentra una puerta con un extraño signo grabado en ella. Consulto su significado en el “Tenebrarum” y este me desvela que es un símbolo del príncipe de las tinieblas. Lanzo agua bendita contra el signo, que desaparece tras una nube de humo. Abro la puerta y, haciendo acopio de valentía, entro.

Un húmedo sótano repleto de ataúdes, abiertos y cerrados, me recibe. En lo alto de una escalera hay una puerta cerrada. Abro los ataúdes y los examino, están vacíos. De repente, sin saber como, la puerta del sótano se abre, aunque no hay nadie bajo su marco.

Ante la invitación que me ofrece la puerta abierta, subo por la escalera y me encuentro en el interior del palazzo Balbi. La puerta principal está cerrada. El lujo es propio de épocas pasada. Una escalera lleva a la planta superior, subo por ella.

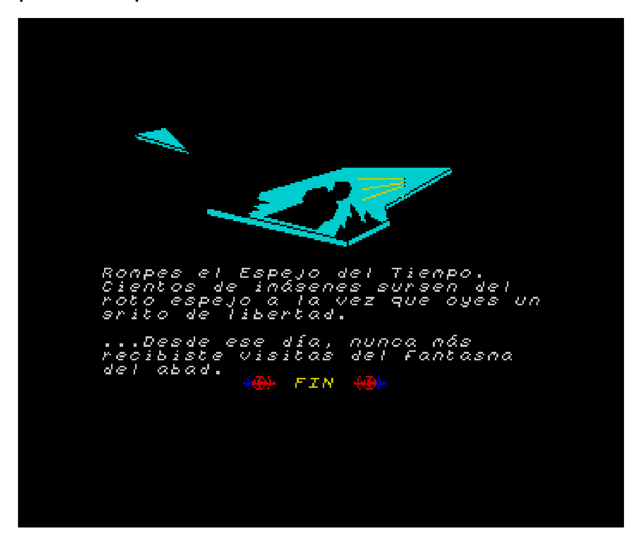
Hellen está allí, junto a Drácula. Un ejército de vampiros me rodea mientras el no-muerto acaricia a mi amada. Drácula saca mi moneda y me reta a un macabro juego. Elijo cara, y pierdo. El príncipe de las tinieblas desaparece con mi amada, los vampiros se abalanzan a por mí. En un rápido movimiento les muestro el crucifijo y se transforman en una bandada de murciélagos volando por encima de la ciudad de Venecia. Es hora de volver a casa.

Cojo la moneda y bajo a la planta principal del palacio Balbi y salgo a la calle. Camino hacia el norte y, tras pasar la iglesia, llego a las puertas de la pensión desde donde viajo de vuelta a Bélgica.



ACTO IV: REDEMPTIO

Unos niños juguetean a la puerta de mi casa, en Vlaming straat. Abro la puerta y entro al vestíbulo. Subo a la primera planta y me dirijo al espejo del tiempo. Sin pensarlo dos veces lo rompo en mil pedazos. Nadie se verá envuelto en maldiciones por su culpa.



Descarga el mapa de la aventura:
mapa_tristesalas.png

MIGUEL

el aventurero

HISTORIA DE LA AVENTURA

Las aventuras de texto o conversacionales fueron un género de un relativo éxito en el final de la época comercial del Spectrum. Antes, la barrera idiomática había sido un impedimento para la llegada de algunas joyas de la programación inglesas, y el bajo interés del público en general (quizá por su excesiva juventud) había hecho que las pocas aventuras que se habían editado en castellano fueran recibidas con cierta frialdad por el mercado español.

El primer intento de crear una aventura comercial en España fue Yength, en 1984, y venía de la mano de la compañía Dinamic. Este primer intento fue un rotundo fracaso por dos razones: el público aún no estaba preparado para este tipo de juegos y además Yength era una aventura de baja calidad, con una jugabilidad muy baja.

Sólo un año más tarde, en 1985, la revista Microhobby, en el número dos de su publicacion compañera "Microhobby Cassete" publicaba la obra de Luis E. Juan "Alicia en el país de las maravillas", basada en la obra de Lewis Caroll y, aunque su exito fue moderado, sin duda fue muy superior al de Yenght, debido a su mayor calidad, pero sobre todo a algo que luego daría gran éxito a otras aventuras: estar basada en una historia conocida.



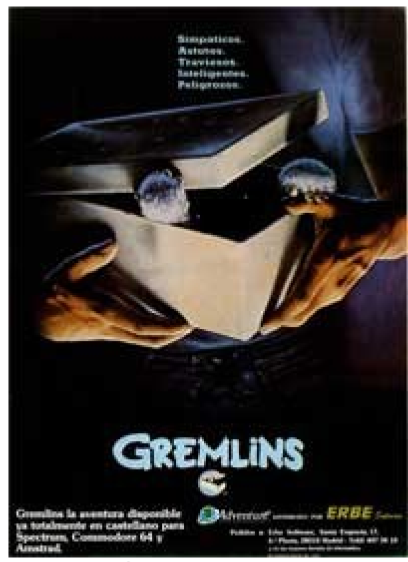
Alicia-MH

Mientras tanto apenas algunas aventuras inglesas llegaban a España, la mayoría de la mano de la piratería, puesto que las distribuidoras no parecían querer arriesgarse a importar aventuras en inglés, y mucho menos a traducirlas. El ejemplo mas claro de esto es The Hobbit, basada en la obra de Tolkien, que aunque había sido publicada por Melbourne House en 1982 no fue hasta 1984 cuando empezó a 'distribuirse' en nuestro pais. "The Hobbit" compartía con "Alicia en el Pais de la Maravillas" el nexo con una historia conocida, y así igual que muchos quisimos ser Alicia en aquel juego, muchos también quisimos ser Bilbo en el juego de Melbourne House, aunque fuera con el diccionario en la mano.



The Hobbit

Una excepción a dicha carencia de importaciones se produjo en 1985: la primera traducción de una conversacional inglesa al castellano (y probablemente la última, al menos oficial): "Gremlins". La aventura basada en la película de igual nombre enganchó por lo conocido del tema, y por una calidad medio-alta, aunque algunos tuvimos que lidiar con la traducción (EMPUJAR BOTON era una construcción castellana digamos que un poco forzada).



Anuncio de Gremlins

En 1986 Dinamic intentaba de nuevo la entrada en el mundo conversacional, de la mano de Arquimedes XXI, de Jorge Blecua, Luis Rodríguez y Javier Cubedo. Esta vez Dinamic cuidó más las formas y Arquimedes XXI, aunque tuvo un éxito moderado, no es recordada como una aventura mala, como pasó con Yength.

También en 1986, Dinamic publicó "Cobras' Arc", una aventura de las entonces llamadas iconográficas, claras precursoras de lo que después vendría con el auge de las aventuras gráficas en PC (Sierra / Lucas Film).

Un año mas tarde, Dinamic volvía a la carga con el que fue el buque insignia del 'boom' aventurero que vendría después: El Quijote. De nuevo una obra conocida, que además estaba siendo emitida en aquel momento a modo de serie de dibujos animados en TVE, facilitó enormemente la aceptación de aquella aventura. Además, fue la primera aventura comercial española que utilizó una herramienta específica para la creación de aventuras en lugar de partir de cero y desarrollar en BASIC o lenguaje máquina. El Quijote utilizaba el conocido parser GAC, de Incentive Software.



El Quijote

Dinamic observó que las ventas de El Quijote habían sido buenas, y que probablemente el esfuerzo de hacer la aventura usando el parser era inferior al de realizar otros juegos, así que visto el filón comenzó a editar diversos títulos: "Megacorp", 1987, "La guerra de las vajillas", 1987, una parodia de la saga Star Wars, "Los pájaros de Bangkok", 1988, basada en las obras de Vázquez Montalbán. Todos ellos fueron de una calidad aceptable y fueron éxitos moderados de ventas.

En 1988 las aventuras ya gozaban por tanto de un moderado éxito, que vino a reforzarse con la aparición de importantes novedades:

En 1988 Andrés Samudio funda, tras un acuerdo con Dinamic, la compañía Aventuras AD. Esto supone el cese de la producción de aventuras por parte de Dinamic, que delega en Aventuras AD dicha línea de negocio y se compromete a actuar como distribuidora de las mismas. Por otro lado, el propio Andrés Samudio comienza a escribir una serie de artículos en la conocida revista Microhobby, denominados "El mundo de la aventura", que acercan al mundillo a gente antes nada interesada en este tipo de juegos. Por último, la propia AD realiza la conversión (y distribución) en castellano del parser más utilizado en aquel momento en Inglaterra: el Professional Adventure Writing System (más conocido como PAW o PAWS), de Gilsoft.



PAWS

Adicionalmente, AD funda el Club de Aventuras AD, un punto de reunión de los aventureros en forma de fanzine, que ayudó a mucha gente a mantener el interés, e incluso a distribuir sus propias obras creadas con PAW. Así, entre 1988 y 1992 se crearon muchas compañías homegrown, que ayudaron a enriquecer el panorama mientras que Aventuras AD editaba sus mejores aventuras:

- Supervivencia (El Firfurcio), 1988
- El Jabato, 1989
- Aventura Original, 1989
- Aventura Espacial, 1990
- La Diosa de Cozumel, 1990
- Los Templos Sagrados, 1991
- Chichen Itza, 1992

De todas ellas, sólo Supervivencia (que en realidad era una pequeña demo de PAWS aparecida en la cinta que acompañaba por aquel entonces a la revista Microhobby) y La Aventura Espacial (un intento extraño de acercar las aventuras conversacionales a las gráficas) no fueron éxitos. El resto de la producción de AD se convirtió en las mejores aventuras en castellano realizadas hasta la fecha.

Pero no fueron las Aventuras de AD las únicas publicadas en esa época. Aprovechando el tirón, algunas otras compañías comerciales editaron sus propias aventuras:

- Abracadabra, 1988, de Odisea Software, publicada por Proein
- La Corona, 1988, de Pedro Amador, publicada por System4
- Zipi y Zape, de Magic Hand



Abracadabra

Por otro lado, entre 1987 y 1988, y hasta 1994, comenzaron a existir como ya se ha dicho diversas compañías homegrown que distribuían sus obras por medio del correo, fanzines y algunas otras publicaciones amateur, y no podemos olvidarnos de ellas puesto que entre ellas están algunas de las obras de más calidad de la época (aparte de la mayor parte de la producción, por encima de la comercial).

Son tantísimas las obras y tantas las compañías que sería imposible nombrarlas todas, pero quisiera recordar especialmente algunas de ellas: Year Zero Software S.p.A. (con autores como Fran Morell, Jon), Grupo Creators Union (Josep Coletas), Wazertown Works (Carlos Sisí), JSJ Soft (Javier San José) o Errata Choft (Juan Antonio Paz). Si queréis conocer más aventuras de la época es altamente recomendable la visita al Proyecto BASE.

Se han mencionado los fanzines: durante la época de la publicacion de los artículos de Samudio en Microhobby, e incluso algunos años después del cese de la publicación de la revista, los fanzines mantuvieron unido el mundillo, así el fanzine de CAAD, el fanzine Zeta For Zero de Year Zero Software, "El Aventurero", "CPAC" o "A través del espejo" reunieron entre sus páginas las aventuras (comerciales o homegrown, españolas o inglesas) del momento.



Los vientos del Walhalla, Grupo Creators Union

En 1992 la muerte comercial del Spectrum era un hecho. Las compañías no producían juegos para Spectrum y las revistas del sector había emigrado a otros campos. La escena aventurera del Spectrum aguantaría aún otros dos años, con algunas producciones homegrown, pero en 1995 la producción prácticamente había cesado o se había trasladado al mundo PC y Amiga.

Desde entonces hasta ahora el mundo aventurero ha producido algunas obras para Spectrum, pero en general la producción se ha dirigido a otras máquinas, y últimamente sobre todo a máquinas virtuales, como vimos en el número 12 de Magazine ZX. No obstante, como se vio en aquel artículo, es posible jugar a algunas de estas aventuras modernas en nuestros Spectrum +3 y, además, recientemente el retorno de uno de los autores homegrown más prolíficos de los 80/90 (Josep Coletas) nos está dotando de nuevas aventuras hechas directamente para Spectrum como son la saga del Dr. Van Hallen y la reciente "Código Secreto Lucybel: Cryogenic". Son 6 aventuras en apenas un año, la más alta tasa de producción de juegos de Spectrum desde el 94, aventuro.

Y a partir de hoy todo la producción de juegos para Spectrum depende de ti: PAWS aún está ahí y ha demostrado ser capaz aun de dar grandes obras que compiten con obras para PC, y si prefieres el C siempre puedes seguir el curso presentado por Siew en números anteriores de esta revista.

LINKS

Proyecto BASE: http://www.speccy.org/base, bolsa de aventuras de Spectrum en Español.

Club de aventuras AD: http://caad.mine.nu, punto de reunión de los 'aventureros' actuales.

UTO

programación z88dk

LA LIBRERIA SPRITE PACK

En artículos anteriores de la revista hemos visto como utilizar las funciones proporcionadas de base con la librería z88dk para la creación de sprites, aplicándolas para crear la semilla de lo que podría ser nuestro propio juego de coches. Sin embargo, nos encontrábamos con un par de contratiempos que, aunque no se indicaron de forma explícita, los podríamos descubrir a la hora de crear nuestras propias aplicaciones. Por una parte, la complejidad del diseño de los propios sprites; de hecho, nos teníamos que basar en una aplicación (sólo para Linux) para la creación de los mismos. Por otra parte, cambiar los atributos de color y fondo de los sprites, así como transparencias, es un tema no especialmente sencillo que ni siquiera se trató.

Para hacernos la vida más fácil a la hora de programar usando sprites disponemos de la librería `Sprite Pack`; la página principal del proyecto es:

http://www.geocities.com/aralbrec/

Se trata de piezas de código escritas en ensamblador, con una interfaz que puede ser usada desde C para su compilación con las herramientas de z88dk. Esto en cristiano quiere decir que podremos llamar a las funciones de la librería Sprite Pack desde nuestros programas z88dk de forma totalmente transparente, pero teniendo la seguridad de que el código será lo más eficiente posible.

Sprite Pack es una librería multipropósito, no centrada únicamente en la creación de sprites. Se compone de varios módulos distintos: creación de sprites, rutinas de entrada, intersecciones de rectángulos, tipos abstractos de datos, compresión de datos, API de interrupciones, etc. No solo podemos utilizarla con el Spectrum, sino que en teoría podríamos emplear todo este código en cualquier plataforma Z80, aunque existen dos módulos demasiado específicos que solo dispondremos para Spectrum y Timex, el de creación de sprites y el de lectura de dispositivos de entrada.

Durante los siguientes artículos nos centraremos especialmente en el módulo de sprites, con la idea de ser utilizado en el seno de nuestras aplicaciones z88dk para Spectrum, aunque durante el desarrollo de nuestros ejemplos tendremos que ir usando también algunas pequeñas partes del resto de los módulos. Vamos a dar un pequeño paso atrás con respecto a entregas anteriores, pues vamos a aprender de nuevo a dibujar sprites desde el principio, aunque utilizando Sprite Pack en esta ocasión. Podría parecer que esto es un poco inútil, pero ya veremos más adelante que las ventajas de utilizar esta capa por encima de z88dk no tardarán en llegar.

INSTALACIÓN

Lo primero que debemos hacer es descargar la librería de su página (en el momento de escribir este artículo, la última versión era la 2.2):

http://www.geocities.com/aralbrec/

Tras obtener el fichero comprimido, lo descomprimimos en cualquier directorio de trabajo. El resultado será la carpeta z88dk, en cuyo interior encontraremos a su vez otro directorio llamado work, en el que en último lugar descubriremos un nuevo directorio llamado splib2, en el que nos deberemos situar para realizar la compilación.

Al tratarse de una librería para el manejo, entre otras cosas, de sprites de la pantalla, debemos comprobar que el código de la librería se va a compilar para las características peculiares del Spectrum antes de continuar. Lo único que tendremos que hacer será editar el fichero *SPconfig.def*, que es el fichero de configuración de la compilación, y asegurarnos de que el valor de

DISP_SPECTRUM es 1 y el valor del resto de variables DISP_ (DISP_HICOLOUR, DISP_HIRES y DISP_TMXDUAL) es 0.

Sprite Pack incluye un fichero Makefile.bat que creará la librería en una máquina Windows. Solamente sería necesario en este tipo de entorno que se abriera una ventana de MS-DOS, se acudiera al directorio donde está el código fuente, y tras inicializar las variables de entorno que permiten compilar con z88dk, teclear Makefile. En Linux es un poco más complicado, y deberemos modificar ese archivo para que funcione. Para ahorrar trabajo al lector, se incluye un fichero Makefile.sh, equivalente al Makefile.bat para Windows, que puede ser usado en Linux junto a este artículo. No se debe olvidar el proporcionar permiso de ejecución a dicho fichero para que lo podamos utilizar. El fichero sp.lst, que se encuentra en el directorio raíz del código fuente de la librería (junto a makefile.bat) también debe ser modificado para Linux; junto a este artículo se adjunta igualmente este archivo para poder ser utilizado en este sistema.

Sea cual sea el sistema operativo que utilicemos, como resultado obtendremos un nuevo fichero splib2.lib, que deberemos copiar al interior del directorio lib\clibs dentro de nuestro directorio de instalación de z88dk. A su vez, deberemos copiar el fichero spritepack.h en el directorio include, también dentro del directorio de instalación de z88dk. ¡Ya tenemos la librería lista para ser usada!. Solo será necesario añadir la línea #include "spritepack.h" en cada uno de los archivos .c en los que queramos hacer uso de las funcionalidades de la librería, y compilarlos añadiendo -lsplib2 como uno de los parámetros de zcc.

NUESTRO PRIMER EJEMPLO

Sprite Pack tiene una forma particular de actualizar la pantalla. Realizar una actualización completa y simultánea de toda ella simultáneamente haría necesario el uso de rutinas en ensamblador demasiado específicas, lo cual chocaría con el objetivo real de la librería, que se pretende que sea multiplataforma. Es por ello que lo que realmente se hace es actualizar únicamente las porciones de la pantalla donde se ha producido algún cambio. Se dirá que aquellas partes de la pantalla que queramos redibujar deberán ser invalidadas. Aquellas partes que no queramos redibujar serán regiones validadas. Luego lo veremos con un ejemplo y lo entenderemos mejor, pero de momento nos debemos hacer a la idea de que nunca podremos hacer juegos con scroll total, ya sea horizontal o vertical, con esta librería. Deberemos centrarnos en juegos de plataformas o juegos limitados donde toda la acción se desarrolla sin scroll.

Otros conceptos que deberemos tener claros son el de backtile y sprite. Comprender la diferencia entre ambos es necesario para desarrollar nuestras aplicaciones. La pantalla está dividida en un array de tamaño 32x24 de celdas de caracteres (a su vez de tamaño 8x8). Cada una de estas celdas podrá contener sólo un backtile y uno o más sprites. Los backtiles se pueden entender como el "fondo", y en realidad se trata de UDGs coloreados de tamaño 8x8, que son escritos en la pantalla de la misma forma en la que lo son los UDGs en BASIC. Por otra parte, los sprites podrán ocupar cualquier posición de la pantalla, en incrementos de un pixel, y su tamaño podrá ser mayor que 8x8, aunque no tendremos tanta libertad como con z88dk y únicamente podrán tener tamaños múltiplos, tanto en número de filas como de columnas, de 8.

De momento nos centraremos en los backtiles. La función básica para dibujarlos es sp_PrintAtInv, que además de dibujar el carácter con el color de tinta y papel que deseemos, invalidará la celda donde haya sido dibujado para que se redibuje al actualizar la pantalla. Existe otra función, sp_PrintAt, que hace exactamente lo mismo, pero sin invalidar la celda donde el backtile es dibujado. Todos los cambios en la pantalla se realizarán de forma simultánea al llamar a la función sp_updateNow.

Veamos un ejemplo. El siguiente código se encarga de dibujar letras 'x' en posiciones al azar de la pantalla, cambiando el color de la tinta y el papel, también al azar.

```
#include
#include

#pragma output STACKPTR=61440

main()
{
   #asm
   di
   #endasm
   sp_InitIM2(0xf1f1);
   sp_CreateGenericISR(0xf1f1);
   #asm
```

```
    ei
    #endasm


    sp_Initialize(INK_BLACK | PAPER_WHITE, ' ');
    while(1) {
            sp_UpdateNow();
            if (rand()%10 > 5)
                    sp_PrintAtInv(rand()%24, rand()%32, INK_RED |
PAPER_CYAN, 'x');
            else
                    sp_PrintAtInv(rand()%24, rand()%32, INK_CYAN |
PAPER_RED, 'x');
            sp_Pause(20);
    }
}
```

Veamos línea por línea de qué va todo esto. Las dos primeras se corresponden con las sentencias #include necesarias; en este caso la librería estándar y el archivo de cabecera de la librería Sprite Pack.

La sentencia `#pragma` deberemos incluirla siempre al principio de nuestros programas escritos para la librería Sprite Pack, antes del método main. Lo que se dice con ella es que se desea comenzar la ejecución del programa con el puntero de la pila del Z80 apuntando a la dirección 61440. Al hacerlo de esta forma, evitaremos que la función `sp_initialize` destruya cierta información de la memoria.

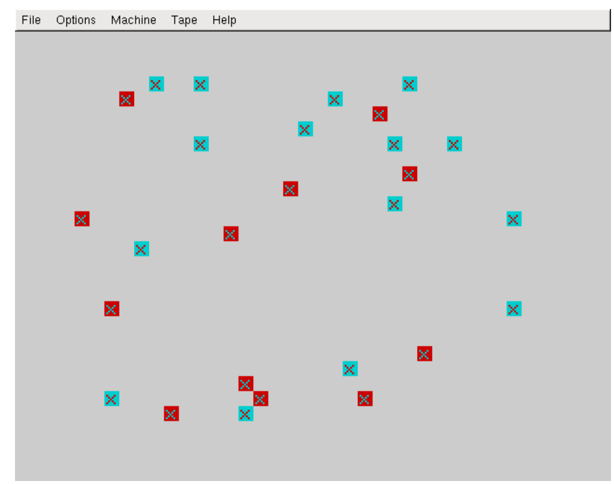
Las primeras líneas de código en el interior de main(), entre el primer `#asm` y el último `#endasm` tendremos que ponerlas también siempre en nuestros programas que usen Sprite Pack. No es necesario entrar en detalle, pero diremos que este código tiene algo que ver con deshabilitar interrupciones para que ciertas funciones como sp_Invalidate funcionen.

Y por fin comienzan las líneas interesantes. La función sp_Initialize es necesaria para que el módulo de sprites de la librería comience a funcionar. En este caso, los parámetros hacen que la pantalla se inicialice escribiendo espacios en blanco, con color de fondo (paper) blanco y tinta negra (ink).

A continuación nos introducimos en el bucle principal, que se va a ejecutar de forma ininterrumpida por toda la eternidad (a menos que detengamos la ejecución del programa). La estructura es muy sencilla. Primero se llama a `sp_UpdateNow` para que se redibujen las celdas de la pantalla donde hubo algún cambio. A continuación se obtiene un número al azar entre 1 y 10 (mediante la función `rand()`), y en función de su valor llamaremos a `sp_PrintAtInv` con unos parámetros u otros.

El método `sp_PrintAtInv` dibuja un backtile en la pantalla e invalida la posición donde dicho backtile ha sido colocado, para que la celda correspondiente sea redibujada tras llamar a `sp_UpdateNow`. Los dos primeros parámetros indican la posición del backtile (que también los obtenemos al azar), a continuación el color de papel y de tinta (con la misma sintaxis que en el caso de la función `sp_Initialize`) y por último indicamos el carácter a escribir. Este carácter se corresponde con un UDG, como en BASIC. De momento no hemos asociado ningún UDG a la letra x, por lo que se mostrará en pantalla será la letra x apareciendo en posiciones al azar, teniendo o bien color de tinta rojo y papel cyan o al revés.

La última línea del bucle, `sp_Pause`, introduce un pequeño retardo. Actúa exactamente igual que el comando PAUSE de BASIC.



Espectaculares efectos gráficos conseguidos con la librería Sprite Pack

NUESTRO PRIMER EJEMPLO CON UN SPRITE

En esta sección vamos a dibujar un sprite simple sobre un fondo un poco más complejo. Para comprender cómo podemos crear un fondo más complejo, hemos de recordar que los backtiles pueden entenderse como si fueran UDGs de BASIC; sin embargo, hasta ahora sólo hemos utilizado caracteres alfanuméricos. ¿De qué forma definimos gráficos para estos backtiles?

Mediante la función `sp_TileArray` asociamos un determinado UDG 8x8 a un carácter, de tal forma que al escribir dicho carácter como backtile en la pantalla, aparecerá su UDG correspondiente. Esta función recibe como parámetro el carácter al que queremos asociar el UDG y un array de tipo uchar con la definición del UDG. Ese array contendrá un valor hexadecimal por cada fila del UDG, utilizando la misma notación que vimos en capítulos anteriores para la creación de sprites de tamaño 8x8 con z88dk. Por ejemplo, observemos el siguiente código:

```
uchar fondo[] = {0x55,0xaa,0x55,0xaa,0x55,0xaa,0x55,0xaa};

sp_TileArray(' ', fondo);
sp_Initialize(INK_WHITE | PAPER_BLACK, ' ');
```

Con la primera línea creamos un array llamado fondo que contendrá la definición de un UDG de tipo "tablero de ajedrez" (un píxel negro, un píxel blanco, un píxel negro, un píxel blanco, y así sucesivamente). Con la instrucción sp_TileArray asociamos este UDG al carácter de espacio en blanco, de tal forma que cuando en la línea siguiente llamamos a sp_Initialize, el fondo se llenará de copias del UDG asociado a dicho carácter. De esta forma tan sencilla podremos tener fondos más detallados.

¿Y cómo definimos un sprite? De forma mucho más fácil a la vista en artículos anteriores, en los que se usaba z88dk sin ningún añadido. Nada más simple que una notación como la siguiente:

```
#asm

._sprite1
defb @00111100, @11000011
defb @01000010, @10000001
defb @10000001, @00000000
defb @10100101, @00000000
defb @10000001, @00000000
defb @10011001, @00000000
defb @01011010, @10000001
defb @00111100, @11011011

#endasm
```

Como se puede observar, volvemos a hacer uso de código ensamblador empotrado en el interior de nuestro código C, entre las directivas `#asm` y `#endasm`. En este caso concreto estamos definiendo un único sprite 8x8, para el cual necesitaremos dos bloques de bits de ese tamaño. El de la izquierda es el dibujo en sí mismo del sprite, que en nuestro caso en una pequeña carita con la boca abierta. El bloque de la derecha indica las transparencias; en aquellas posiciones donde el valor sea 1, el sprite será transparente y se verá lo que haya en el fondo. Evidentemente, es necesario que dichas posiciones tengan un valor 0 en el bloque de la izquierda.

A este bloque de datos que marcan un sprite de 8x8 le hemos llamado sprite1. Este nombre nos servirá para referenciar este código ensamblador desde el código C. Para ver cómo dibujar el sprite en la pantalla, nada mejor que un pequeño ejemplo:

```
#include
#pragma output STACKPTR=61440

extern struct sp_Rect *sp_ClipStruct;
#asm
LIB SPCClipStruct
._sp_ClipStruct         defw SPCClipStruct
#endasm


extern uchar sprite1[];
uchar fondo[] = {0x55,0xaa,0x55,0xaa,0x55,0xaa,0x55,0xaa};
```

```
void *my_malloc(uint bytes)
{
            return sp_BlockAlloc(0);
}


void *u_malloc = my_malloc;
void *u_free = sp_FreeBlock;



main()
{
            struct sp_SS *bicho;


            #asm
            di
            #endasm
            sp_InitIM2(0xf1f1);
            sp_CreateGenericISR(0xf1f1);
            #asm
            ei
            #endasm

            sp_TileArray(' ', fondo);
            sp_Initialize(INK_WHITE | PAPER_BLACK, ' ');
            sp_Border(BLACK);
            sp_AddMemory(0, 255, 14, 0xb000);

            bicho = sp_CreateSpr(sp_MASK_SPRITE, 1, sprite1, 1,
TRANSPARENT);
            sp_MoveSprAbs(bicho, sp_ClipStruct, 0, 10, 15, 0, 0);

            sp_UpdateNow();

            while(1);
}

#asm

._sprite1
defb @00111100, @11000011
defb @01000010, @10000001
defb @10000001, @00000000
defb @10100101, @00000000
defb @10000001, @00000000
defb @10011001, @00000000
defb @01011010, @10000001
defb @00111100, @11011011

#endasm
```

Comentemos línea por línea el programa. La sentencia #pragma ya la conocemos del ejemplo anterior, así que no hace falta que nos detengamos en ella. A continuación se define una estructura de la forma siguiente:

```
extern struct sp_Rect *sp_ClipStruct;
#asm
LIB SPCClipStruct
._sp_ClipStruct         defw SPCClipStruct
#endasm
```

Esta estructura tan confusa está sacada de spritepack.h y define un rectángulo que cubre toda la superficie de la pantalla. Existen muchas definiciones como ésta dentro de dicho fichero de cabecera. Para qué sirven lo veremos mucho más adelante, cuando tratemos el tema de las colisiones. De momento deberemos saber que lo necesitamos para poder dibujar el sprite en la pantalla.

La línea extern uchar sprite1[]; crea el array que contendrá la información del sprite. Este array se llena con los datos definidos entre las cláusulas #asm y #endasm al final del programa, después de la etiqueta ._sprite1, y contendrá la definición del sprite, de la misma forma en la que en la línea siguiente se crea un array de tipo uchar llamado fondo con la definición de un UDG para asociarlo al backtile de fondo.

El siguiente código:

```
void *my_malloc(uint bytes)
{
           return sp_BlockAlloc(0);
}


void *u_malloc = my_malloc;
void *u_free = sp_FreeBlock;
```

entra dentro de lo que estaremos obligados a insertar en nuestro programa cada vez que queramos añadirle sprites, aunque no entendamos muy bien de qué se trata. Se obtiene a partir del módulo de manejo de memoria de Sprite Pack y sirve para que el programa pueda obtener memoria bajo demanda. Esto es necesario debido a la forma en que la pantalla es actualizada al usar esta librería. Además de las estructuras creadas por el programador, la librería creará otras propias durante la ejecución y la memoria para dichas estructuras se obtendrá por medio de esta función.

Y por fin comienza el método main. En primer lugar definimos la variable que va a contener el sprite que vamos a mostrar por pantalla. Esta variable es de tipo struct sp_SS, una estructura que contiene diversa información sobre un sprite, como su localización y su tamaño. A continuación, entre las directivas #asm y #endasm, el código ensamblador necesario en cada uno de nuestros programas Sprite Pack del que hablamos en la sección anterior.

Y continuamos con las dos líneas siguientes, cuyo funcionamiento ha sido explicado anteriormente:

```
p_TileArray(' ', fondo);
sp_Initialize(INK_WHITE | PAPER_BLACK, ' ');
```

La línea posterior es nueva, y permite definir el color del borde de la pantalla (como la instrucción BORDER de BASIC). A continuación otra línea un tanto complicada, correspondiente a la llamada a la función sp_AddMemory, con la que se reserva memoria para los sprites. En concreto estamos reservando 255 bloques de 14 bytes a partir de la dirección 0xb000 que se sabe que está libre. Para cada sprite deberíamos reservar un par de bloques de 14 bytes, por lo que 255 es algo desproporcionado para nustro ejemplo. Si nos vemos apurados de memoria podemos reservar menos en este paso.

Las líneas que más nos interesan de este ejemplo son las siguientes:

```
bicho = sp_CreateSpr(sp_MASK_SPRITE, 1, sprite1, 1, TRANSPARENT);
sp_MoveSprAbs(bicho, sp_ClipStruct, 0, 10, 15, 0, 0);
```

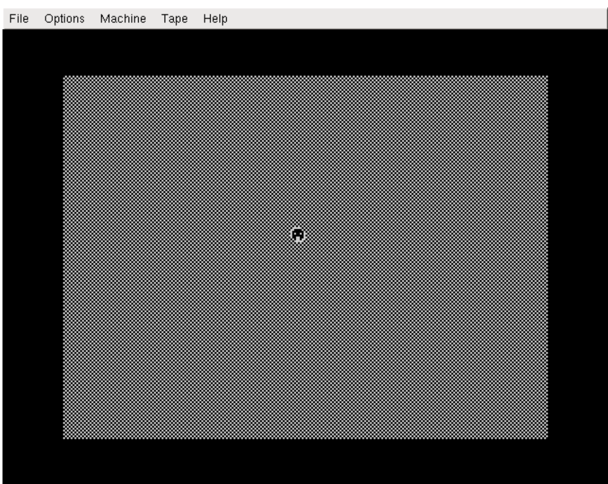
Con la primera de ellas creamos el sprite en si mismo. El resultado de la llamada se almacenará en la variable de tipo struct sp_SS bicho declarada anteriormente, que usaremos en el resto de métodos con los que queramos manipular dicho sprite. El primer parámetro (para el que nosotros hemos usado MASK, pero que podría ser XOR, OR o LOAD) indica la forma en la que el sprite se dibuja en la pantalla. El tipo MASK es el más lento de dibujar, y utiliza una máscara para determinar que porciones del sprite serán transparentes (tal como se ha visto anteriormente). Los tipos OR y XOR se corresponderían con los ya vistos en artículos anteriores. El siguiente parámetro indica el número de rejillas 8x8 que forman el sprite (se hablará de esto más adelante). A continuación, el nombre del array de tipo uchar que contiene la definición del sprite. En tercer lugar, el plano que ocupará el sprite en pantalla. El valor de este parámetro podrá valer entre 0 y 63. Cuanto mas bajo sea este valor más cerca del usuario se encontrará el sprite, de tal forma que sprites con

valores bajos del tercer parámetro serán situados "encima" de sprites con valores altos, tapándolos. El cuarto parámetro tiene que ver con el color, y lo veremos también más adelante. Hemos usado el valor TRANSPARENT para que no influya en el color del fondo.

Situaremos el sprite creado sobre la pantalla mediante el uso de la función `sp_MoveSprAbs`, que desplaza nuestro personaje a una posición absoluta de la pantalla. Podríamos utilizar `sp_MoveSprRel`, que acepta el mismo número de parámetros, con la diferencia de mover el sprite a una posición relativa a la actual. Por lo tanto, siempre usaremos sp_MoveSprAbs para colocar al sprite en su posición inicial.

Como primer parámetro, el sprite a mover. Como segundo parámetro, el rectángulo que hace referencia a toda la pantalla definido anteriormente. El tercer parámetro hace referencia a la animación, y tampoco hablaremos de él en esta ocasión. El cuarto y el quinto, al bloque de la pantalla donde situaremos el sprite. Decíamos anteriormente que la pantalla estaba dividida en bloques de 8x8; pues bien, con estos dos parámetros indicamos la celda donde colocaremos nuestro sprite. Para conseguir una colocación más exacta, podemos usar los dos últimos parámetros, correspondientes al offset. Indican cuantos píxeles mover hacia la derecha y hacia abajo, respectivamente, a partir de la celda cuya posición es la indicada por los dos parámetros anteriores.

Por último, con `sp_UpdateNow()` redibujamos las porciones de la pantalla que lo necesiten, y con while(1) dejamos nuestro programa en constante ejecución, hasta el resto de los días, o hasta que lo detengamos. El resultado se puede observar en la siguiente captura de pantalla.



Un terrible ser intergaláctico espera ansioso la orden de atacar

UN SPRITE GRANDE Y NERVIOSO

En este apartado veremos un pequeño ejemplo que introduce algunos conceptos interesantes, como la creación de sprites de un tamaño superior a 8x8 y el uso del movimiento relativo. Vamos a hacer aparecer una criatura en nuestra pantalla que se mueva de forma nerviosa y aleatoria.

Pero comencemos por el principio. ¿Cómo definimos sprites que contengan más de un bloque 8x8? Con Sprite Pack, los sprites grandes se definen por columnas. Podemos definir una columna de un sprite grande, compuesta por uno o más bloques de 8x8, bajo una misma etiqueta de código ensamblador. Un sprite de más de una columna se formará a partir de varias de estas definiciones en ensamblador.

Por ejemplo, mostramos como quedaría un sprite con un tamaño de dos bloques de alto por dos bloques de ancho listo para ser usado con Sprite Pack:

```
#asm

._bicho1
defb @00000011, @11111100
defb @00000100, @11111000
defb @00001000, @11110000
defb @00001011, @11110000
defb @00001011, @11110000
defb @00001000, @11110000
defb @00001000, @11110000
defb @00000100, @11111000

defb @00000011, @11111100
defb @00001100, @11110011
defb @00001100, @11110011
defb @00011000, @11100111
defb @00011000, @11100111
defb @01111100, @10000011
defb @01111100, @10000011
defb @00000000, @11111111

._bicho2
defb @11100000, @00011111
defb @00010000, @00001111
defb @00001000, @00000111
defb @01101000, @00000111
defb @01101000, @00000111
defb @00001000, @00000111
defb @10001000, @00000111
defb @10010000, @00001111

defb @11100000, @00011111
defb @00011000, @11100111
defb @00011000, @11100111
defb @00001100, @11110011
defb @00001100, @11110011
defb @00111110, @11000001
defb @00111110, @11000001
defb @00000000, @11111111

#endasm
```

Como se puede observar, definimos dos columnas para el sprite, cada una de ellas formada a su vez por dos sprites de tamaño 8x8, incluyendo su máscara de transparencias. Para utilizar el sprite en el código deberíamos hacer algo similar a esto:

```
extern uchar bicho1[];
extern uchar bicho2[];

main()
{
        struct sp_SS *spriteBicho;

         spriteBicho = sp_CreateSpr(sp_MASK_SPRITE, 2, bicho1, 1,
TRANSPARENT);
         sp_AddColSpr(spriteBicho, bicho2, TRANSPARENT);
         sp_MoveSprAbs(spriteBicho, sp_ClipStruct, 0, 10, 15, 0, 0);
}
```

Se debe hacer uso, en primer lugar, de `sp_CreateSpr` para asignar la primera columna del sprite a la variable de tipo `struct sp_SS`, y en segundo lugar, de `sp_AddColSpr`, tantas veces como columnas adicionales debamos añadir. En este caso, se ha usado un valor de 2 para el segundo parámetro de `sp_CreateSpr`, indicando que cada columna del sprite tendrá un tamaño de dos bloques de 8x8. Efectivamente, este segundo parámetro indica el número de bloques que tendrá cada columna de nuestro sprite; por eso en nuestro primer ejemplo le dimos valor 1. Una vez se ha creado el sprite con `sp_CreateSpr` y se han añadido las columnas correspondientes con `sp_AddColSpr`, se podrá tratar la estructura sp_SS resultante como un todo, tal como se demuestra en la llamada a `sp_MoveSprAbs` que sigue a las dos líneas anteriores.

Es importante destacar que todas las columnas de un mismo sprite deben de ser definidas de forma contigua en la memoria. Esto se traduce en que tenemos que definirlas de forma contigua también en nuestro código.

Sin embargo, hay algo que hasta ahora no hemos tenido en cuenta, y es debido a que no hemos movido nuestros sprites por la pantalla. Cuando trasladamos sprites usando el píxel y no el bloque como unidad de medida (los dos últimos parámetros de sp_MoveSprAbs y sp_MoveSprRel servían para esto) veremos como los sprites no son correctamente dibujados; solo se redibuja la parte del sprite más a la izquierda que cabe dentro de una misma celdilla de la pantalla. Un truco para evitar esto es crear sprites un poco más anchos y más altos de lo que realmente necesitamos. Para ello, añadimos una nueva columna en blanco, y en cada columna, un nuevo bloque en blanco al final. En el caso concreto de nuestro sprite 2x2 anterior, deberíamos definirlo de esta forma:

```
#asm

._bicho1
defb @00000011, @11111100
defb @00000100, @11111000
defb @00001000, @11110000
defb @00001011, @11110000
defb @00001011, @11110000
defb @00001000, @11110000
defb @00001000, @11110000
defb @00000100, @11111000

defb @00000011, @11111100
defb @00001100, @11110011
defb @00001100, @11110011
defb @00011000, @11100111
defb @00011000, @11100111
defb @01111100, @10000011
defb @01111100, @10000011
defb @00000000, @11111111
```

```
defb @00000000, @11111111
defb @00000000, @11111111
defb @00000000, @11111111
defb @00000000, @11111111
defb @00000000, @11111111
defb @00000000, @11111111
defb @00000000, @11111111
defb @00000000, @11111111

._bicho2
defb @11100000, @00011111
defb @00010000, @00001111
defb @00001000, @00000111
defb @01101000, @00000111
defb @01101000, @00000111
defb @00001000, @00000111
defb @10001000, @00000111
defb @10010000, @00001111
```

```
defb @11100000, @00011111
defb @00011000, @11100111
defb @00011000, @11100111
defb @00001100, @11110011
defb @00001100, @11110011
defb @00111110, @11000001
defb @00111110, @11000001
defb @00000000, @11111111

defb @00000000, @11111111
defb @00000000, @11111111
defb @00000000, @11111111
defb @00000000, @11111111
defb @00000000, @11111111
defb @00000000, @11111111
defb @00000000, @11111111
defb @00000000, @11111111
```

```
._bicho3
defb @00000000, @11111111
defb @00000000, @11111111
defb @00000000, @11111111
defb @00000000, @11111111
defb @00000000, @11111111
defb @00000000, @11111111
defb @00000000, @11111111
defb @00000000, @11111111

defb @00000000, @11111111
defb @00000000, @11111111
defb @00000000, @11111111
defb @00000000, @11111111
defb @00000000, @11111111
defb @00000000, @11111111
defb @00000000, @11111111
defb @00000000, @11111111

defb @00000000, @11111111
defb @00000000, @11111111
defb @00000000, @11111111
defb @00000000, @11111111
defb @00000000, @11111111
defb @00000000, @11111111
defb @00000000, @11111111
defb @00000000, @11111111

#endasm
```

Por lo tanto, nuestro sprite 2x2 se convierte en un sprite 3x3 al añadir una nueva columna a la derecha y un nuevo bloque en la parte inferior de cada columna. Nuestro código a la hora de usar el sprite debería ser en este caso algo más parecido a esto:

```
extern uchar bicho1[];
extern uchar bicho2[];
extern uchar bicho3[];

main()
{
         struct sp_SS *spriteBicho;

         spriteBicho = sp_CreateSpr(sp_MASK_SPRITE, 3, bicho1, 1,
TRANSPARENT);
         sp_AddColSpr(spriteBicho, bicho2, TRANSPARENT);
         sp_AddColSpr(SpriteBicho, bicho3, TRANSPARENT);
         sp_MoveSprAbs(spriteBicho, sp_ClipStruct, 0, 10, 15, 0, 0);
}
```

Como queda patente, indicamos que el tamaño en bloques de cada columna es 3 al llamar a la función `sp_CreateSPr`, y además se llama dos veces a `sp_AddColSpr` para añadir la segunda y la tercera columna a nuestro bicho.

A continuación se muestra un ejemplo completo, con nuestro sprite moviéndose al azar por la pantalla, por medio de la función `sp_MoveSprRel`:

```
#include
#include
#pragma output STACKPTR=61440

extern struct sp_Rect *sp_ClipStruct;
#asm
LIB SPCClipStruct
._sp_ClipStruct         defw SPCClipStruct
#endasm

extern uchar bicho1[];
extern uchar bicho2[];
extern uchar bicho3[];
uchar hash[] = {0x55,0xaa,0x55,0xaa,0x55,0xaa,0x55,0xaa};

void *my_malloc(uint bytes)
{
           return sp_BlockAlloc(0);
}

void *u_malloc = my_malloc;
void *u_free = sp_FreeBlock;
```

```
main()
{
           char dx, dy, i;
           struct sp_SS *spriteBicho;

            #asm
            di
            #endasm
            sp_InitIM2(0xf1f1);
            sp_CreateGenericISR(0xf1f1);
            #asm
            ei
            #endasm

           sp_TileArray(' ', hash);
           sp_Initialize(INK_WHITE | PAPER_BLACK, ' ');
           sp_Border(CYAN);
           sp_AddMemory(0, 255, 14, 0xb000);


           spriteBicho = sp_CreateSpr(sp_MASK_SPRITE, 3, bicho1, 1,
TRANSPARENT);
           sp_AddColSpr(spriteBicho, bicho2, TRANSPARENT);
           sp_AddColSpr(spriteBicho, bicho3, TRANSPARENT);
           sp_MoveSprAbs(spriteBicho, sp_ClipStruct, 0, 10, 15, 0, 0);

           while(1) {
                 sp_UpdateNow();

                 dx = dy = 1;

                if (rand()%2 == 0) // izquierda
                        dx = -dx;
                else if (rand()%2 == 0) // derecha
                        dx = 0;
                if (rand()%2 == 0) // arriba
                        dy = -dy;
                else if (rand()%2 == 0) // abajo
                        dy = 0;

                sp_MoveSprRel(spriteBicho, sp_ClipStruct, 0, 0, 0, dx,
dy);
           }
}

#asm

defb @00000000, @11111111
defb @00000000, @11111111
defb @00000000, @11111111
defb @00000000, @11111111
defb @00000000, @11111111
defb @00000000, @11111111
defb @00000000, @11111111

._bicho1
defb @00000011, @11111100
defb @00000100, @11111000
defb @00001000, @11110000
defb @00001011, @11110000
defb @00001011, @11110000
defb @00001000, @11110000
defb @00001000, @11110000
defb @00000100, @11111000
```

```
defb @00000011, @11111100
defb @00001100, @11110011
defb @00001100, @11110011
defb @00011000, @11100111
defb @00011000, @11100111
defb @01111100, @10000011
defb @01111100, @10000011
defb @00000000, @11111111

defb @00000000, @11111111
defb @00000000, @11111111
defb @00000000, @11111111
defb @00000000, @11111111
defb @00000000, @11111111
defb @00000000, @11111111
defb @00000000, @11111111
defb @00000000, @11111111

._bicho2
defb @11100000, @00011111
defb @00010000, @00001111
defb @00001000, @00000111
defb @01101000, @00000111
defb @01101000, @00000111
defb @00001000, @00000111
defb @10001000, @00000111
defb @10010000, @00001111

defb @11100000, @00011111
defb @00011000, @11100111
defb @00011000, @11100111
defb @00001100, @11110011
defb @00001100, @11110011
defb @00111110, @11000001
defb @00111110, @11000001
defb @00000000, @11111111

defb @00000000, @11111111
defb @00000000, @11111111
defb @00000000, @11111111
defb @00000000, @11111111
defb @00000000, @11111111
defb @00000000, @11111111
defb @00000000, @11111111
defb @00000000, @11111111

._bicho3
defb @00000000, @11111111
defb @00000000, @11111111
defb @00000000, @11111111
defb @00000000, @11111111
defb @00000000, @11111111
defb @00000000, @11111111
defb @00000000, @11111111
defb @00000000, @11111111

defb @00000000, @11111111
defb @00000000, @11111111
defb @00000000, @11111111
defb @00000000, @11111111
defb @00000000, @11111111
defb @00000000, @11111111
defb @00000000, @11111111
defb @00000000, @11111111
```

```
defb @00000000, @11111111
defb @00000000, @11111111
defb @00000000, @11111111
defb @00000000, @11111111
defb @00000000, @11111111
defb @00000000, @11111111
defb @00000000, @11111111
defb @00000000, @11111111

#endasm
```

Ya deberíamos entender la práctica totalidad de este código, que mueve un sprite al azar por la pantalla, por lo que tan solo haremos dos apuntes:

- Los movimientos indicados con sp_MoveSprRel son movimientos relativos, por lo que si queremos desplazarnos tan solo un píxel, ya sea a la izquierda o a la derecha, ya sea arriba o abajo (penúltimo y último parámetros respectivamente) usaremos valores +1 y -1.
- Se ha añadido un bloque vacío antes de definir nuestro sprite, al final del código anterior; esto es así porque debemos asegurarnos de que haya un bloque en blanco encima de cada columna (manías del ensamblador generado por Sprite Pack).

¿Y AHORA QUÉ?

Hemos aprendido las más básicas funcionalidades de Sprite Pack para la creación de sprites. Ya somos capaces de crear nuestros propios sprites (de forma más sencilla a la que se ha realizado en artículos anteriores) y de moverlos, aunque sea de manera aleatoria, por la pantalla (sin necesidad de hacer llamadas de ensamblador para el manejo de interrupciones, como en el ejemplo de los coches visto en el número anterior de MagazineZX). Por lo tanto, ganamos en facilidad de uso.

Vemos que no es necesario borrar un sprite usando la máscara XOR antes de volver a dibujarlo para simular movimiento, tal como hacíamos con z88dk sin Sprite Pack, y también observamos como somos capaces de añadir transparencias en los sprites de forma muy simple. Volvemos a ganar en facilidad de uso.

En el siguiente artículo añadiremos colores a nuestros sprites y aprenderemos a moverlos con el teclado. También veremos como borrar de forma efectiva un sprite. Si el espacio lo permite (y si no es así, que no cunda el pánico, pues se verá en números posteriores de la revista) codificaremos un simple juego funcional con todo lo aprendido.

LINKS

Código fuente

SIEW

El terrible ser intergaláctico se ha hecho mayor y es más nervioso que antes...



ENTREVISTA A JOSEP COLETAS CAUBET

Aunque el género de las aventuras conversacionales pueda ser minoritario, paradójicamente es uno de los más prolíficos en desarrollos de videojuegos actuales para ZX Spectrum. De entre los diversos autores destaca Josep Coletas y su excelente saga con el doctor Van Halen como protagonista, quien gustosamente nos contesta a esta entrevista.

¿Quién es Josep Coletas y cómo se inicia con las aventuras conversacionales?

Escritor de aventuras, músico, astrónomo, egiptólogo, informático, etc, etc.

La primera aventura que jugué fue el Hobbit, y me quedé bastante impresionado (aunque no puedo negar que tuve mucha ayuda de una revista que había por aquellos tiempos llamada ZX). Cuando hice SINDBAD el tratamiento de la pantalla era idéntico a The Hobbit (paper blanco, una línea de GDUs separando descripción de las acciones...). Eso basta para decir que me influyó bastante. Luego vino un par de secciones dirigidas por un tal Andrés Samudio en una revista llamada MICRO-HOBBY. Compramos el PAW Castellano y todo comenzó.

Previamente al hecho de ser creador, fuiste usuario de aventuras, ¿cuáles eran/son tus favoritas?

Pues de las inglesas me encantó Kayleth, y españolas la Aventura Original. Hay más pero seguro que me olvidaría de alguna.

"Hacer las aventuras aprovechando las posibilidades del PC haría perder toda la gracia artesanal del asunto."

Antes de crear el sello Grupo Creators Union, hiciste algunas aventuras que no llegaron a ver la luz, ¿qué impresión tienes ahora de aquellas primeras producciones?

Supongo que te refieres a SINDBAD, TIEMPOS DE MAGIA y GOLLUMITIS. Pues la misma que entonces. No eran más que pruebas para comprobar un poco qué se podía hacer con el PAWs, y como tales deben ser tomadas. Respecto a SINDBAD, era una aventura hecha en BASIC inspirada en unos dibujos animados japoneses (nada que ver con el Sindbad de la Disney), por lo que al estar realizada en dicho lenguaje era muy limitada.

¿Cómo decides un día hacer una aventura conversacional creando el sello Grupo Creators Union? ¿Quienes lo formabais y qué aventuras creasteis?

Pues a partir de que compré el PAWS. Aunque yo me encargaba del trabajo duro, tenía amigos que colaboraban en algunas cosas, como aportando documentación (en aquellos tiempos no tenía internet así que toda aportación documental era a base de libros), comprobando errores ortográficos, echar alguna partida para testear, etc.

Las aventuras oficiales del Grupo Creators Union son las siguientes (por orden de creación): El Forastero, Pueblo de la Noche, El Señor del Dragón, La Prehistoria, Idiliar y Los Vientos del Walhalla.

¿Cómo era el proceso de desarrollo de una aventura y con qué equipos, tanto hardware como software, contabais?

Pues disponíamos de un ZX Spectrum Plus y un ZX Spectrum + 2 A.

Aquí tenéis detallados los pasos para crear "Los Vientos del Walhalla":

1. Buscar información sobre el tema de la aventura para crear el argumento.
2. Crear el argumento cuidando al máximo

que no haya nada que quede flojo o traído por los pelos.
3. Pensar qué información acompañará al juego para que el jugador pueda jugar sin problemas la aventura.
4. Hacer borrones y croquis de todo lo que aparece en la aventura. Hasta que se tenga todo bastante claro.
5. Empezar a programar la aventura de la siguiente forma:

   A) En una hoja (o más hojas) se escribirá el número de bandera y a su lado qué función realiza.
   B) En otra hoja (o más hojas) se escribirá el número de proceso y a su lado qué función realiza.
   C) Lo mismo con los Mensajes y si se acaban pues también con los del Sistema.
   D) Lo mismo con las Localidades.
   E) Lo mismo con los gráficos de las localidades y pictures.
   F) Lo mismo con el Vocabulario. Ayudarse con diccionarios de sinónimos.
   G) Cada vez que se termina una parte de la aventura hay que grabar la base de datos que se llamará "Nombre de la aventura1". Y cuando se cargue y se realice una nueva grabación se llamará "Nombre de la aventura2" y luego 3, y 4, etc. (En LOS VIENTOS DEL WALHALLA llegamos hasta la 115 aproximadamente).

Una de vuestras aventuras, el Señor del Dragón, fue presentada al Concurso de Aventuras de la revista MicroHobby donde quedó entre las 15 primeras. ¿Qué os supuso al grupo ese concurso y conseguir ese puesto?

Pues sin duda fue un reconocimiento de nuestro trabajo muy apreciado.

Bajo el nombre de Grupo Creators Union seguisteis sacando varias aventuras más, ¿se distribuyeron a través de la bolsa del CAAD? ¿Llegasteis a recibir alguna oferta de publicación comercial por alguna de ellas?

A través de la Bolsa del CAAD se distribuyeron únicamente 3 aventuras: El Forastero, Pueblo de la Noche y El Señor del Dragón. Luego el director del CAAD de entonces (Juan José Muñoz Falcó) nos comunicó amablemente los pasos para distribuir nosotros mismos nuestras aventuras a través de la venta por correo (en aquel momento la Bolsa estaba llena). Respecto a la oferta de publicación comercial simplemente no (me temo que si querías ver alguna aventura tuya en las tiendas eras tú quien tenía que ir detrás de las distribuidoras, no al revés, y es que el mundillo de la aventura siempre se le ha tenido en un "punto y aparte" dentro de los videojuegos), aunque algún que otro fanzine estaba interesado en nuestras aventuras para sus propias bolsas, por ejemplo creo recordar que con el fanzine "El Aventurero" acordamos la distribución de Idiliar mucho más tarde.



Pueblo de la Noche, una de las primeras aventuras del Grupo Creators Union

En su momento también realizaste una aventura para IBM/PC llamada Barbarian Quest, ganadora de varios premios, pero, ¿por qué no seguiste desarrollando para este ordenador?

Estamos hablando de 1995, y para aquél entonces ya habían pasado muchas cosas desde los inicios de Grupo Creators Union. Así que en 1996 archivé el próximo proyecto ("Los Extraordinarios Casos del Dr. Van Halen", que de hecho en un principio era una sola aventura y bastante distinta a lo que habéis jugado, ya que para entonces tenía poca cosa planeada. Y sí, obviamente no se llamaba "Los Extraordinarios Casos del Dr. Van Halen") y dejé de producir aventuras.

Actualmente has vuelto, ya en solitario, con la estupenda saga del Dr. Van Halen. ¿De dónde surge la idea de crear un saga con este protagonista y exclusiva, por ahora, para ZX Spectrum?

Pues recuperé el proyecto de 1996 antes mencionado, le puse el nombre "Los Extraordinarios Casos del Dr. Van Halen", cambié el escenario (en un principio era la catedral de Notredame de París) añadí a Hallen, el bastón con cuchilla, la biblioteca secreta, la moneda Iuramentum, el Tenebrarum, y un infinito etc., e hice el primer caso para ZX Spectrum simplemente para pasar el rato. Como gustó y me recomendaron que hiciera una versión PC, hice dicha versión PC del primer caso. Pero tras comprobar que las versiones PC me llevaban demasiado tiempo y trabajo para algo que no pasa de ser un hobby, decidí no complicarme la vida y sacar todo lo demás para Spectrum. Lo cierto es que en un principio quería sacar distintos casos del Dr. Van Halen que fueran sencillitos argumentalmente y en desarrollo, pero ya que el primer caso gustó tanto al final decidí esforzarme

también en los demás.

Al haber estado realizadas con el PAWS, ¿es posible su aparición en el resto de plataformas de 8 bits?

No.

El uso de emuladores y programas para IBM/PC facilitarán enormemente el desarrollo con respecto a cuando sólo contabais con el ZX Spectrum, ¿cómo ha cambiado esto a la hora de hacer nuevas aventuras aunque la plataforma final sea el ZX Spectrum?

Pues todas las aventuras del Dr. Van Halen y la aventura Código Secreto Lucybel: Cryogenic han sido realizadas al estilo de "la vieja usanza", es decir, usando el emulador con el PAWs cargado como si de un Spectrum real se tratara y aprovechando únicamente las posibilidades del emulador respecto al tiempo de carga y grabación y otras facilidades similares. De momento no he utilizado programas externos de PC, pero quizás lo haga para mejorar las pantallas de carga (hasta ahora realizadas con el propio PAWs). Y es que me parece bien aprovechar las posibilidades propias del emulador para según qué, pero de eso a hacer las aventuras aprovechando las posibilidades del PC se perdería toda la gracia artesanal del asunto.

¿Cómo ves actualmente el mundo de la aventura conversacional? ¿Hay suficientes adeptos como para seguir creando nuevas aventuras?

Bueno yo hago las aventuras por el placer de hacerlas, y es que a estas alturas no te lo puedes tomar de otra forma, al menos en mi opinión.

¿A qué crees que se debe la escasa proliferación del género de las aventuras conversacionales ya no sólo en las consolas (debido esencialmente a las ausencia de teclado) sino incluso en los PCs?

Pues porque cuando se habla de videojuegos se habla de algo que entra por los ojos, por los oídos y hoy en día, en tres dimensiones y con un buen framerate. Por lo que el poco halagador panorama aventurero siempre ha sido "el poco halagador panorama aventurero" (vamos que esto dudo que llegue a cambiar nunca).

¿Veremos aventuras conversacionales (tuyas o ajenas) que se aprovechen de la pantalla táctil que implementan algunas consolas o agendas personales? ¿Opinas que esto puede mejorar la situación de las aventuras conversacionales en estos dispositivos sin teclado?

Ni idea.



La aventura que inició toda una saga

¿Eres de la opinión de que la aventura gráfica es la evolución natural de la aventura conversacional o no tienen nada que ver?

La aventura gráfica ha sido la forma de acercar las aventuras a la gente que de otro modo nunca hubiera jugado una. Pero claro, estamos hablando de aventuras gráficas, por lo que tampoco es correcto decir que "han jugado una aventura conversacional" (tienen que ver pero son cosas distintas)

Imaginamos que tu afán de desarrollar aventuras no ha acabado y que en el futuro nos sorprenderás con más aventuras, ¿nos puedes adelantar algo de lo que estés haciendo o pensando hacer para el 2006?

Pues me temo que no puedo decir ni prometer nada, ya que todo depende de si tengo tiempo (y ganas).

Por curiosidad, ¿por qué las aventuras conversacionales son los únicos videojuegos que te preguntan si quieres seguir jugando y se resetean en caso contrario?

Pues no tengo ni idea (lo cierto es que no pasaría nada si el juego volviera al principio en lugar de preguntar eso. ¿A lo mejor se hizo así porque los modelos Spectrum 48k carecían de reset?).

Gracias por tu tiempo, Josep, no te robamos más para que sigas desarrollando nuevas aventuras. Tan sólo si quieres comentar algo, este es tu espacio...

Gracias a vosotros por la entrevista y por dedicar un número de MagazineZX a las aventuras conversacionales. Por cierto, seguid sacando MagazineZX también en formato PDF, que os queda de maravilla.

HORACE

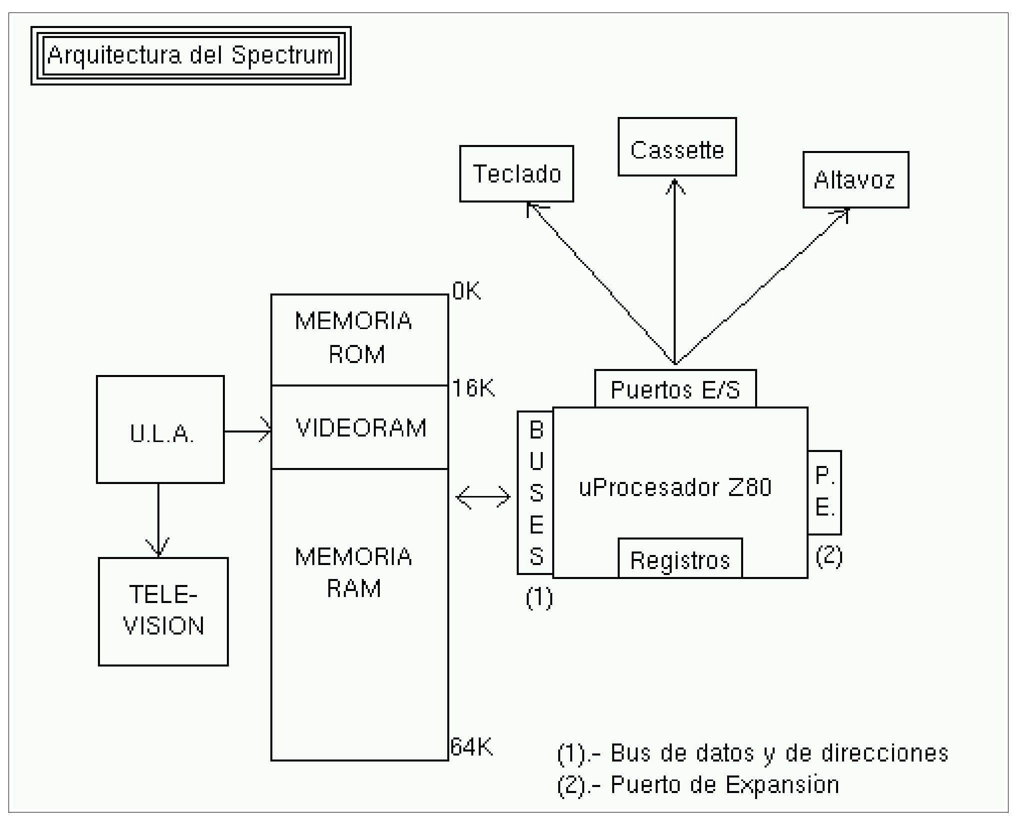
programación ensamblador

ARQUITECTURA Y FUNCIONAMIENTO DEL SPECTRUM

LA ARQUITECTURA DEL SPECTRUM

Antes de comenzar a programar el Spectrum necesitamos conocer su arquitectura: ¿qué hay dentro de nuestro pequeño ordenador y cómo funciona? En BASIC muchas veces podemos olvidarnos de los detalles a nivel de hardware (para eso es un lenguaje de Alto Nivel), pero en ensamblador no: al hablar directamente un lenguaje que se traduce a código máquina, necesitamos conocer exactamente cómo funciona internamente.

Aquí veremos una visión simplificada de la arquitectura hardware del Spectrum pero que en el fondo es todo lo que necesitaremos para la mayoría de programas. Para empezar veremos un esquema de cómo es internamente nuestro Spectrum a nivel de hardware, y después comentaremos uno a uno los elementos que lo componen:



Esquema del hardware de un ZX Spectrum

En un vistazo general, podemos ver que el microprocesador Z80 se conecta mediante los puertos de entrada/salida de la CPU a los periféricos externos (teclado, cassette y altavoz de audio), pudiendo leer el estado de los mismos (leer del teclado, leer del cassette) y escribir en ellos (escribir en el altavoz para reproducir sonido, escribir en el cassette) por medio de estas

conexiones conocidas como "I/O Ports".

Al mismo tiempo, los Buses de Datos y de Direcciones conectan al microprocesador con la memoria. Esta conexión es la que permite que el Z80 pueda leer y escribir en cualquier posición de la memoria. Cuando encendemos el Spectrum, lo que éste lee de la memoria son instrucciones. Empezando por la posición 0000 (0 Kb), el Spectrum comienza a leer instrucciones y a ejecutarlas, una a una. En la primera parte de la memoria tenemos la ROM del Spectrum, que contiene instrucciones de programa pre-programadas y que no podemos modificar: es el menú del Spectrum y el intérprete de BASIC.

Por otro lado, nuestro microprocesador tiene una serie de registros internos con los que trabaja y que son los que manipula y utiliza para ejecutar las instrucciones almacenadas en la memoria.

Algo muy importante sobre la memoria es que hay una zona de ella que se conoce como videomemoria. Es memoria RAM normal y corriente, sólo que los datos que contiene son leídos por un chip llamado ULA muchas veces por segundo, y conforman la imagen que vemos en el televisor de nuestro Spectrum. Escribiendo un valor en una de estas direcciones de memoria (poniendo a 1 uno de sus bits), veremos aparecer en el televisor un punto. La ULA es, pues, el chip encargado de representar en el televisor el contenido de la videomemoria y nosotros, cuando queramos escribir o dibujar algo en pantalla, ya no utilizaremos funciones como PLOT o DRAW, sino que escribiremos directamente valores en esta zona de memoria.

Por último, el puerto de expansión del Spectrum permite conectar nuevos periféricos (como el adaptador de Joystick Kempston o el Interface 1 ó 2) directamente a las patillas de la CPU, ampliando las funcionalidades del ordenador.

Veamos más detalladamente los diferentes componentes de la arquitectura del Spectrum, y cómo funcionan.

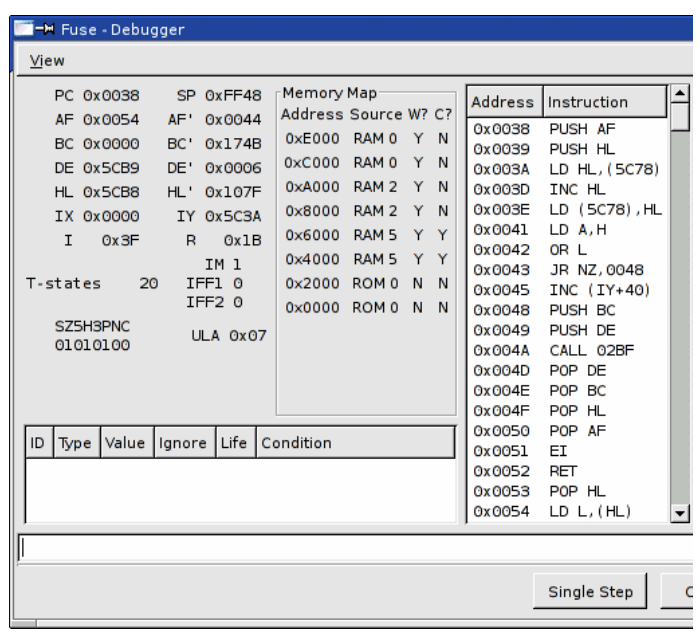
EL MICROPROCESADOR Z80

Como podemos distinguir en el esquema, el cerebro de nuestro Spectrum es un microprocesador Zilog Z80 a 3,54Mhz. Un microprocesador es un circuito integrado que consta (principalmente) de registros, microcódigo, puertos de entrada/salida, un bus de datos y uno de direcciones.



Imagen de un C.I. Z80 de Zilog

Los registros son variables (igual que cualquier variable de BASIC) que residen dentro de la misma CPU. En el caso del Z80, tiene 2 juegos de registros con unos nombres concretos: entre otros, lo forman registros de un byte como A, F, B, C, D, E, H, L, I y R, y los registros de dos bytes IX, IY, SP y PC. Veremos los registros en detalle en su momento (así como el segundo juego de registros disponible), pero podemos hacernos a la idea de que los registros son simples variables de 8 ó 16 bits que utilizaremos en nuestros programas en ensamblador. Así, podremos cargar un valor en un registro (LD A, 25), sumar un registro con otro (ADD A, B), activar o desactivar determinados bits de un registro (SET 7,A), etc.



Parte del debugger de FUSE mostrando los registros de la CPU

El juego de registros es todo lo que tenemos (aparte de la memoria) para realizar operaciones en nuestro programa: siempre que estemos operando con datos o utilizando variables, tendrá que ser por fuerza un registro, o una posición de memoria que usemos como variable. Por ejemplo, podemos escribir el siguiente programa en ensamblador, que sumaría dos números:

```
LD A, 10
LD B, 20
ADD A, B
```

El anterior programa, una vez ensamblado y ejecutado en un Z80, vendría a decir:

- Carga en el registro A el valor "10".
- Carga en el registro B el valor "20".
- Suma el valor del registro A con el del registro B y deja el resultado en el registro A (A=A+B).

Tras ejecutar el anterior programa en un Z80, el contenido del registro A sería 30 (10+20). Cuando comencemos a explicar las diferentes instrucciones del Z80 veremos en más detalle los registros, su tamaño, cómo se agrupan, y de qué forma podemos usarlos para operar entre ellos y realizar nuestras rutinas o programas.

Existe un registro especial llamado PC (Program Counter, o Contador de Programa). Este registro de 16 bits puede contener un valor entre 0 y 65535, y su utilidad es la de apuntar a la dirección de memoria de la siguiente instrucción a ejecutar. Así, cuando arrancamos nuestro Spectrum, el registro PC vale 0000h, con lo que lo primero que se ejecuta en el Spectrum es el código que hay en 0000. Una vez leído y ejecutado ese primer código de instrucción, se incrementa PC para apuntar al siguiente, y así continuadamente. Los programas se ejecutan linealmente mediante (como veremos) un ciclo basado en: Leer instrucción en la dirección de memoria apuntada por PC, incrementar registro PC, ejecutar instrucción. Posteriormente veremos más acerca de PC.

Ya hemos visto lo que son los registros del microprocesador. Ahora bien, en el ejemplo anterior, ¿cómo sabe el microprocesador qué tiene que hacer cuando se encuentra un comando "LD" o "ADD"? Esto es tarea del microcódigo. El microcódigo del microprocesador es una definición de qué tiene que hacer el microprocesador ante cada una de las posibles órdenes que nosotros le demos.

Por ejemplo, cuando el microprocesador está ejecutando nuestro anterior programa y lee "LD A, 10" (en realidad, lee de la memoria los opcodes 62 y 10), el Z80 utiliza el microcódigo encargado de mover el valor 10 al registro A. Este microcódigo no es más que una secuencia de señales hardware y cambios de estados electrónicos cuyo resultado será, exactamente, activar y desactivar BITs en el registro A (que no es más que una serie de 8 biestables electrónicos que pueden estar a 0 voltios o a 5 voltios cada uno de ellos, representando el estado de los 8 bits del registro A). Lo mismo ocurrirá cuando se lea la instrucción "LD B, 20", sólo que se ejecutará otra porción de microcódigo que lo que hará será modificar el registro B.

Este microcódigo está dentro del microprocesador porque sus diseñadores implementaron todas y cada una de las operaciones que puede hacer el Z80. Cuando pedimos meter un valor en un registro, leer el valor de un registro, sumar un registro con otro, escribir el valor de un registro en una dirección de memoria, saltar a otra parte del programa, etc, para cada una de esas situaciones, hay un microcódigo (implementado mediante hardware) que realiza esa tarea. Nosotros no tendremos que preocuparnos pues de cómo hace el Z80 las cosas internamente a nivel de microcódigo, aunque es bueno que conozcáis cómo llega el Spectrum a ejecutar nuestros comandos: gracias al microcódigo.

PUERTOS DE ENTRADA/SALIDA

Si cogemos un microprocesador Z80, podremos distinguir muchas patillas de conexión. Además de las patillas que se utilizan para alimentar el Z80 desde la fuente de alimentación, existen otra serie de patillas para "Puertos de Entrada/Salida" y "Bus de datos y de direcciones". Esas patillas son la conexión del microprocesador con el resto de elementos del ordenador.

<--	A11	1	Z80 CPU	40	A10	-->
<--	A12	2		39	A9	-->
<--	A13	3		38	A8	-->
<--	A14	4		37	A7	-->
<--	A15	5		36	A6	-->
-->	CLK	6		35	A5	-->
<->	D4	7		34	A4	-->
<->	D3	8		33	A3	-->
<->	D5	9		32	A2	-->
<->	D6	10		31	A1	-->
	Vcc	11		30	A0	-->
<->	D2	12		29	GND	
<->	D7	13		28	!RFSH	-->
<->	D0	14		27	!M1	-->
<->	D1	15		26	!RESET	<--
-->	!INT	16		25	!BUSRQ	<--
-->	!NMI	17		24	!WAIT	<--
<--	!HALT	18		23	!BUSAK	-->
<--	!MREQ	19		22	!WR	-->
<--	!IORQ	20		21	!RD	-->

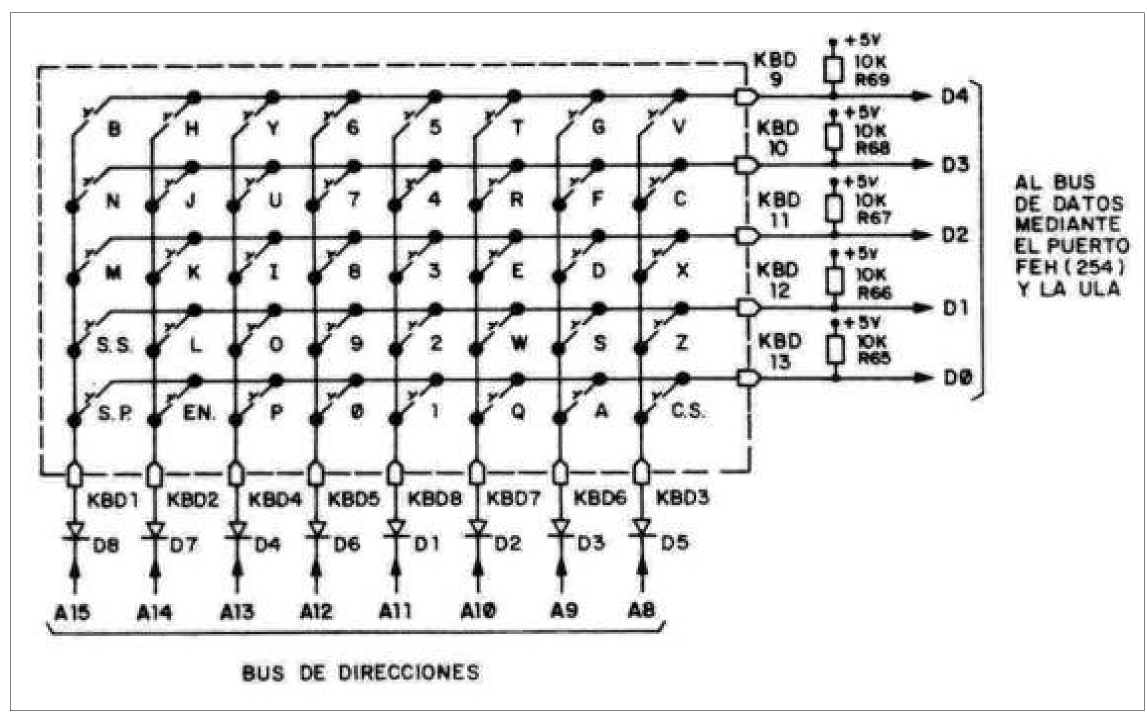
Las 40 patillas del microprocesador Z80. donde vemos el bus de direcciones (A0 a A15) y el bus de datos (D0 a D7), que se conectan a los puertos de entrada/salida y a la memoria

Los registros del microprocesador y el microcódigo son internos al procesador, y ninguna de las patillas que podemos ver en la imagen anterior nos comunica con ellos. Es el procesador (internamente) quien lee/modifica los registros o quien ejecuta microcódigo cuando nosotros se lo decimos con las instrucciones de nuestro programa.

Pero nuestro procesador, como hemos dicho, necesita conectarse con elementos del exterior, entre ellos (y casi exclusivamente en el caso del Spectrum) tenemos la memoria, el teclado, el altavoz y la unidad de cinta.

Visto de un modo muy simple, el Z80 puede acceder a través de una serie de patillas hasta a 65536 elementos externos. Esos elementos se conectan a la CPU Z80 a través de las patillas de "Buses de datos y direcciones" (que podemos leer en los puertos de Entrada/Salida). Una vez conectados (la conexión se realiza físicamente a nivel de hardware, con pistas en la placa base), el microprocesador Z80 puede leer los valores que el dispositivo pone en esas pistas, o puede escribir valores en esas pistas para que el dispositivo los utilice.

Supongamos el ejemplo del teclado: un teclado es (a un nivel muy simple) una matriz de pulsadores. Al final de toda esa matriz de pulsadores, lo que acabamos teniendo es un número de 8 bits (0-255) cuyos bits nos dicen la teclas pulsadas. Pues bien, ese número de 8 bits es lo que el mismo teclado pone en los "cables" que lo unen con el Z80. El teclado se conecta a la CPU mediante una conexión a uno de los puertos (una serie de patillas de datos y direcciones) del microprocesador y gracias a esto el micro-procesador (y nuestro programa) puede leer en todo momento el estado del teclado (y saber qué teclas están pulsadas y cuales no) leyendo del puerto correspondiente.



Matriz de teclado del Spectrum y conexiones al Z80

Así, en nuestros programas podemos leer el estado del teclado mediante (por ejemplo):

```
5  REM Mostrando el estado de la
   fila 1-5 del teclado
10 LET puerto=63486
20 LET V=IN puerto: PRINT AT 20,0;
   V ; "  " : GO TO 20
```

Este ejemplo lee (en un bucle infinito) una de las diferentes filas del teclado, mediante la lectura del puerto 63486 (F7FEh) con la instrucción de lectura de puertos de BASIC "IN". El estado de las 5 teclas desde el 1 hasta el 5 del teclado del Spectrum está conectado a 5 "hilos" de los 8 que llegan al puerto 63486 del micro Z80. Cuando se pulsa una tecla, el teclado pone a 0 (a 0 voltios) el "hilo" correspondiente de esos 8, y nosotros podemos conocer el estado de la tecla leyendo dicho puerto y mirando el bit en cuestión. Al ejecutar el ejemplo anterior, veremos que la pulsación de la tecla "1" cambiará el valor que aparece en pantalla. Si pasamos los diferentes valores que aparecen a binario y nos fijamos en el estado de los 5 últimos bits, nos daremos cuenta como al pulsar y soltar las diferentes teclas del 1 al 5 estaremos variando esos bits entre 0 (al pulsarlas) y 1 (al liberarlas). Los 3 bits más altos del byte debemos ignorarlos, ya que no tienen relación con el teclado.

Si no hay ninguna tecla pulsada, los 5 bits más bajos del byte que hay en el puerto estarán todos a 1, mientras que si se pulsa alguna de las teclas

del 1 al 5, el bit correspondiente a dicha tecla pasará a estado 0. Nosotros podemos leer el estado del puerto y saber, mirando los unos y los ceros, si las teclas están pulsadas o no.



Pulsando "1", ponemos a 0 el bit 0, pasando de 191 a 190

Así, leyendo del puerto 63486 obtenemos un byte cuyos 8 bits tienen como significado el estado de cada una de las teclas de la semifila del "1" al "5".

Bits	D7	D6	D5	D4	D3	D2	D1	D0
Teclas	XX	XX	XX	"5"	"4"	"3"	"2"	"1"

Los bits D7 a D5 no nos interesan en el caso del teclado (por ejemplo, D6 tiene relación con la unidad de cinta), mientras que los bits de D5 a D0 son bits de teclas (0=pulsada, 1=no pulsada). Tenemos pues el teclado dividido en filas de teclas y disponemos de una serie de puertos para leer el estado de todas ellas:

Puerto	Teclas
65278d (FEFEh)	de CAPS SHIFT a V
65022d (FDFEh)	de A a G
64510d (FBFEh)	de Q a T
63486d (F7FEh)	de 1 a 5 (and JOYSTICK 1)
61438d (EFFEh)	de 6 a 0 (and JOYSTICK 2)
57342d (DFFEh)	de P a Y
49150d (BFFEh)	de ENTER a H
32766d (7FFEh)	de (space) a B

A la hora de leer estos puertos, el bit menos significativo (D0) siempre hace referencia a la tecla más alejada del centro del teclado ("1" en nuestro ejemplo), mientras que el más significativo de los 5 (D5) lo hace a la tecla más cercana al centro del teclado.
En ensamblador también hay disponibles 2 instrucciones para leer el contenido de un puerto de Entrada/Salida y para escribir un valor en un puerto determinado, las instrucciones se llaman igual que en BASIC: IN y OUT:

```
; Cargamos en BC el valor del puerto a leer
 LD BC, F7FEh

; Leemos en el registro A el valor del puerto
 IN A, (C)
```

Queríamos mostraros el ejemplo del teclado, aunque estamos todavía empezando y puede haber sido algo complicado de entender, por diferentes motivos:

- El primero, explicar qué son los puertos de E/S y cómo están conectados a nivel de hardware con el microprocesador. Como habéis visto, del teclado salen una serie de "hilos" o pistas de circuito impreso que van directamente al Z80, a través de sus diferentes buses y puertos.
- Nosotros podemos, en cualquier momento, leer y escribir en los puertos de E/S. Con esto conseguimos comunicarnos con los periféricos externos. En este caso, podemos leer del teclado (o escribir o leer del cassette, o en el altavoz) con simples operaciones de lectura y escritura.

El lector debería extraer una conclusión extra del ejemplo del teclado: la gran diferencia de proceso que hay entre programar en ensamblador y programar en BASIC. Supongamos que nos interesa leer el estado del teclado para saber si unas determinadas teclas están pulsadas o no. Para esto, en ensamblador (aunque esto también podemos hacerlo en BASIC) leemos directamente el estado del teclado con un par de simples instrucciones (LD + IN). En BASIC, por contra al leer de un INKEY$ estamos esperando la ejecución de un código que, además de leer TODAS las filas del teclado (no sólo aquellas de las teclas que nos interesen), realiza una conversión de todos los bits pulsados o no pulsados mediante una tabla ASCII para al final proporcionarnos el código ASCII de la tecla pulsada. Lo que son varias simples instrucciones IN en ASM, en BASIC se realiza mediante cientos de instrucciones en ensamblador que nos acaban dando la última tecla pulsada. Es por eso que el intérprete BASIC es tan lento: cada operación BASIC son decenas, cientos o miles de instrucciones en ensamblador que nosotros ni vemos ni controlamos. Programando directamente en ASM, el microprocesador hará EXCLUSIVAMENTE lo que nosotros le digamos que haga. He aquí la "mágica" diferencia de velocidad entre ambos lenguajes.

Podéis encontrar más información sobre los puertos de Entrada y Salida en el capítulo 8 sección 32 del manual del +2A y +3, que tenéis

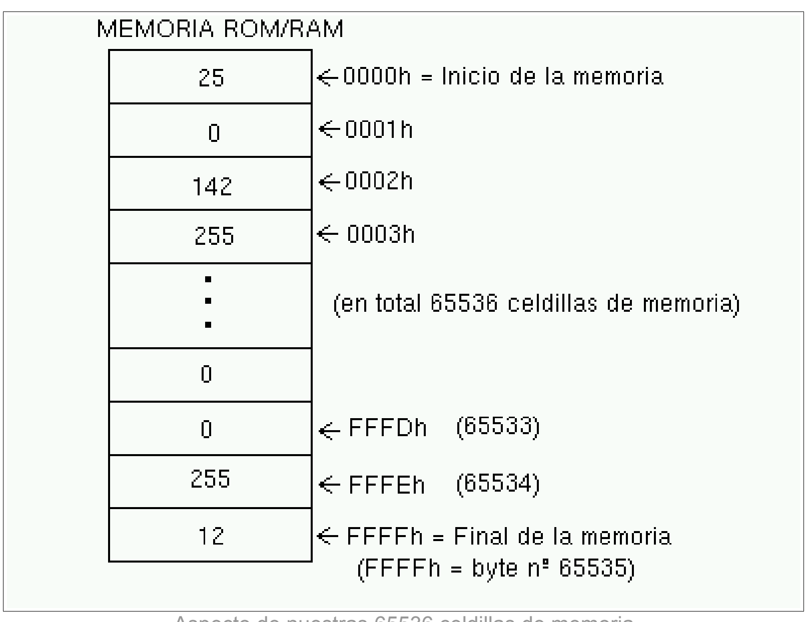
disponible online en World Of Spectrum.

MEMORIA

Al igual que en el caso de los puertos de entrada/salida, nuestro microprocesador está también conectado a los diferentes chips de memoria (hay más de uno). La conexión se realiza siguiendo unas normas definidas por los ingenieros de Sinclair, de tal forma que la memoria se mapea linealmente. ¿Qué quiere decir esto? Que aunque tengamos varios chips de memoria, vemos la memoria como una gran y única memoria de 64KB.

El Spectrum básico (48KB de RAM y 16KB de ROM) tiene disponibles 64KB de memoria, es decir, 65536 bytes a los cuales podemos acceder. Podemos pensar en esta memoria como un gran baúl con 65536 cajones, uno encima de otro. El primer cajón es el cajón 0 (posición de memoria 0), el segundo el cajón 1 (posición de memoria 1), y así hasta el cajón 65535 (posición de memoria 65535).

Nuestro Spectrum no puede tener más de 65536 cajones porque el "bus de direcciones" del microprocesador Z80 es de 16 bits, es decir, las líneas que conectan al microprocesador con la memoria sólo permiten 16 "conexiones"; lo que nos da la posibilidad de acceder a 2 elevado a 16 bytes de memoria, exactamente 65536 bytes.



Aspecto de nuestras 65536 celdillas de memoria

Cada uno de estos cajones (más técnicamente, "celdillas de memoria" o "posiciones de memoria") puede contener un número de 8 bits, con un valor, por tanto, entre 0 y 255. Esto es así porque el "bus de datos" del microprocesador Z80 es de 8 bits, lo que implica que "sólo hay 8 conexiones" entre la salida de datos de la memoria y nuestro procesador.

El microprocesador Z80 puede acceder a cualquier posición de memoria tanto para leer como para escribir. Internamente, cuando le pedimos al microprocesador que meta en el registro A el contenido de la celdilla de memoria 1234h, mediante una instrucción de ensamblador "LD A, (1234h)" (nota, el operador () en ensamblador significa acceso a memoria, y equivaldría en este caso a un "LET A=PEEK 4660" en BASIC) lo que hace el microprocesador internamente es:

Ejecución de LD A, (1234h):

- 1234h en binario es 00010010 00110100, de modo que el Z80 coge las 16 líneas que conectan al microprocesador con la memoria y las pone a esos estados (0 = 0 voltios, 1 = 5 voltios).
- A continuación, el microprocesador utiliza una conexión especial que le conecta con la memoria donde le indica qué operación quiere realizar. Poniendo al valor apropiado la línea de control que le comunica con la memoria, el Z80 informa al chip de memoria de que quiere realizar una operación de lectura.
- La memoria recibe la señal de "quiero leer un dato" por esta señal de control, y mira el bus de direcciones que le conecta con el Spectrum. Mirando el estado de las 16 líneas encuentra el "00010010 00110100" (1234h). Con eso, la memoria sabe a qué "casilla" o "cajón" quiere acceder el microprocesador.
- La memoria lee el valor de la celdilla de memoria 1234h (por ejemplo 0Fh, que es 00001111 en binario) y cambia las 8 conexiones del Bus de Datos para que contengan 00001111 (4 "líneas" las pone a 0 voltios, y las otras 4 a 5 voltios).
- El Z80 consulta el bus de datos y ve el estado de las líneas, con lo que lee el "00001111" o 0Fh.
- El Z80 coloca en el registro A el valor 0Fh.

El procedimiento para escribir es similar, salvo que la línea de control entre el Z80 y la memoria en lugar de indicar "lectura" indica "escritura", y que es el Z80 quien pone en el bus de datos el valor que quiere escribir en la celdilla indicada en el bus de direcciones:

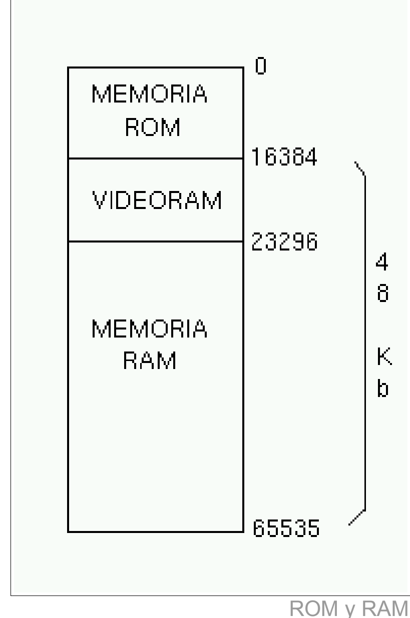
Ejecución de LD (1234h), A

- Supongamos que A contiene el valor 15 (0Fh): el Z80 coloca las líneas del bus de datos a los valores 00001111.
- El Z80 coloca las líneas del bus de direcciones a 00010010 00110100 (1234h).
- A continuación, el microprocesador pone la línea de control READ/WRITE a tal valor que la memoria sabe que el micro le pide una operación de escritura.
- La memoria recibe la señal de "quiero escribir un dato", y mira el bus de direcciones que le conecta con el Spectrum. Mirando el estado de las 16 líneas encuentra el "00010010 00110100" (1234h). Con eso, la memoria sabe a qué "casilla" o "cajón" quiere acceder el microprocesador para escribir.
- La memoria lee el valor del bus de datos para saber qué dato tiene que escribir.
- La memoria escribe en su cajón número 1234h el valor 0Fh.

Estas son las 2 operaciones básicas que el Z80 puede realizar con la memoria: leer una posición de memoria y escribir en una posición de memoria. Nosotros no tenemos que preocuparnos de ninguna de las señales necesarias para realizar lecturas y escrituras, de eso se encarga el microprocesador. Para nosotros, a nivel de ensamblador, nos bastará con ejecutar "LD A, (1234h)" o "LD (1234h), A", por ejemplo.

Las celdillas desde la nº 0 a la 16383 están ocupadas por un chip que es la ROM del Spectrum. Este chip es de sólo lectura (ROM = Read Only Memory), lo cual quiere decir que si intentamos escribir en las celdillas desde la 0 a la 16383 no conseguiremos cambiar el valor almacenado en ellas. ¿Por qué no se puede escribir aquí? Porque es la ROM del Spectrum, es un chip que contiene el sistema operativo del Spectrum, su intérprete BASIC, como veremos posteriormente.

Para trabajar (ejecutar programas, realizar operaciones y tareas) podemos utilizar el resto de la memoria. Desde el cajón o celdilla número 16384 hasta el 65535 podremos escribir y leer.



ROM y RAM

La memoria RAM (celdillas 16384 a 65536) es muy importante para el Spectrum. Para empezar, en ella es donde se almacenan los datos y donde se cargan los programas, y de ella es de donde lee el microprocesador estos programas (como veremos posteriormente) para ejecutarlos instrucción a instrucción. Cuando en nuestra anterior entrega del curso pokeamos la rutina en código máquina en la dirección 40000, estábamos escribiendo un programa en memoria para después ejecutarlo.

Hay una parte de la memoria RAM que es especial. El trozo de 6912 bytes que va desde la dirección 16384 hasta la 23296 es conocida como VideoRAM. Esta porción de la memoria no se utiliza para almacenar programas ni datos, sino que es una representación numérica de los gráficos que aparecen en nuestro televisor. La ULA (un chip que hay dentro de nuestro Spectrum) lee continuamente esta zona de memoria y transforma los unos y ceros que en ella encuentra en puntos y colores en el televisor.

Visto de una manera simple (pero real): al escribir un valor numérico (por ejemplo un 1) en alguna dirección de esta parte de la RAM, de forma inmediata aparece un punto en nuestro televisor, ya que la ULA está continuamente "escaneando" la videoram (de forma independiente del Z80) para reflejar en el televisor todos los valores numéricos que introduzcamos en ella.

Por ejemplo, el siguiente programa pinta 2 píxeles en el centro de la pantalla escribiendo en la videomemoria:

```
10 REM Pintando 2 pixeles en pantalla mediante POKE
20 LET DIRECCION= 16384 + 2000
25 REM 129 = 10000001
30 POKE DIRECCION, 129
40 PAUSE 0
```

Si ejecutamos el programa, veremos 2 puntos en el centro de la pantalla. Estos 2 puntos aparecen al escribir el byte de valor 129 en la dirección 18384. El número 129, en binario, es 10000001. Esos 2 unos son los que se convierten en 2 puntos cuando la ULA lee la videomemoria y transforma los unos en colores de tinta y los ceros en papel.

Lo que nos tiene que quedar claro al respecto de la memoria es lo siguiente:

- El microprocesador puede acceder a la memoria tanto para leer como para escribir, y lo hará cuando nosotros se lo pidamos (para leer o escribir datos en memoria). También lo hará para leer las instrucciones a ejecutar, como veremos posteriormente.
- La memoria está formada por la ROM, la VideoRAM y la RAM.
- En la ROM no podemos escribir (pero sí leer). Almacena el "código" de arranque del Spectrum, así como el intérprete de BASIC. En ella existen rutinas que usa BASIC que podremos aprovechar en nuestros programas en ensamblador.
- En la videoRAM sólo leeremos y escribiremos cuando queramos dibujar cosas en pantalla (puntos y colores).
- En el resto de la RAM (a partir de la dirección 23296) es donde realizaremos todo el trabajo: allí situaremos nuestros programas, nuestras variables, datos, gráficos que después copiaremos a la videoRAM, etc.

EL PUERTO DE EXPANSIÓN

El puerto de expansión del Spectrum no será importante en nuestro curso, ya que no lo utilizaremos para nada. Para vuestra información, basta con conocer que los diferentes pines del puerto de expansión de nuestro ZX están conectados directamente a diferentes patillas del microprocesador. Es decir, cada una de las líneas que podemos ver físicamente en el puerto de expansión es una pista de circuito (como si fuera un cable) que va directamente a alguna de las patillas del Z80.

Es por eso que mediante el puerto de expansión se puede implementar casi cualquier cosa hardware en el Spectrum sin tener que abrirlo: estamos conectando cosas directamente al micro y tras hacerlo, podemos realizar programas que accedan a esos elementos recién conectados: podemos leer los joysticks (porque los estamos conectando a puertos de Entrada/Salida mediante los buses de datos y direcciones), podemos leer cartuchos (porque "ocultamos" la memoria del Spectrum y la reemplazamos con otra memoria que contiene el juego ya cargado en ella, poniendo esta memoria en el Bus de datos y Direcciones), etc.

CÓMO FUNCIONA EL SPECTRUM

Una vez hemos visto todas las partes funcionales, veamos cómo funciona el Spectrum a nivel de ciclo de instrucción.

Para empezar, los diferentes dispositivos externos (teclado, altavoz, cassette, elementos conectados al puerto de expansión, joysticks) se comunican con la CPU por medio de puertos de Entrada/Salida (Puertos E/S o I/O Ports). Para acceder a ellos simplemente leemos o escribimos en el puerto correspondiente mediante las instrucciones IN y OUT del Z80.

Por otra parte, nuestro Z80 puede leer y escribir de la memoria mediante instrucciones LD ("LD valor, (dirección)" y "LD (dirección), valor"), pudiendo utilizar también otro tipo de instrucciones para hacerlo (que veremos en su momento).

En realidad el Z80, visto de una forma simplificada, sólo puede hacer 3 cosas: leer/escribir en la memoria, leer/escribir en los dispositivos de Entrada/Salida y decodificar/ejecutar instrucciones. La parte de lectura/decodificación/ejecución es el funcionamiento principal del microprocesador, y es lo que trataremos a continuación.

CICLO DE EJECUCIÓN DE UN SPECTRUM

Como ya hemos visto, la parte central del Spectrum es un microprocesador Z80 el cual ve la ROM y la RAM de forma continuada como la totalidad de su memoria. Es decir, ve 64KB de memoria de los cuales los primeros 16K son el contenido de chip de ROM, y los siguientes 48K los del chip de RAM. Recordemos que el Spectrum puede leer el contenido de cualquiera de estas 65536 celdillas (así como escribir en ellas, pero sólo a partir de la 16384, ya que los primeros 16KB son de ROM).

Al encender el Spectrum éste se inicializa y muestra el BASIC en pantalla. ¿Por qué ocurre esto? Esto ocurre porque el microprocesador Z80 comienza a ejecutar instrucciones desde la dirección de memoria 0, donde está el principio de la ROM de 16K, es decir, el intérprete BASIC.

Al alimentar con corriente eléctrica el ordenador se ejecuta la ROM: al encender un Spectrum (que perdió en su apagado toda alimentación eléctrica) todos los registros del microprocesador Z80 valen 0 (sin alimentación eléctrica, todos los bits de la CPU están a 0 Voltios, es decir, a 0 lógico), incluído el registro PC (Program Counter o Contador de Programa), que es el que apunta a la siguiente instrucción que el Z80 debe leer y ejecutar. Un microprocesador funciona a grandes rasgos de la siguiente forma:

- Leer instrucción apuntada por el registro PC.
- Incrementar PC para apuntar a la siguiente instrucción.
- Ejecutar la instrucción recién leída.
- Repetir continuadamente los 3 pasos anteriores.

Visto en pseudocódigo, como si fuera un programa (de hecho, es el pseudocódigo de cualquier emulador de Spectrum), un Z80 actúa así:

- Encendido de ordenador:
    - Todos los registros (A, B, C, ..., SP, PC) valen 0.
- Mientras No Se Apague el Ordenador: Leer de la memoria la siguiente instrucción, mediante (contenido de la dirección apuntada por PC):
    - Instruccion = [PC] (Instrucción es un opcode, un número que indica la operación a realizar. Podría ser de más de un byte.)
- PC = PC + 1
- Si la instrucción necesita algún operando, leerlo:
        - Operando1 = [PC]
        - PC = PC + 1
        - Operando2 = [PC] (Opcional)
        - PC = PC + 1 (Opcional)
    - Decodificar la instrucción (mirar en una tabla interna y ver qué microcódigo hay que ejecutar para la instrucción leída).
    - Ejecutar la instrucción

  Fin Mientras

Así pues, al encender el ordenador, PC vale 0. Al estar la ROM mapeada en la posición de memoria 0 (mediante cableado hardware de los chips de memoria en la placa del Spectrum), lo que pasa al encender el ordenador es que ese contador de programa (PC) está apuntando al principio de la ROM, y es por eso que se ejecuta la ROM paso a paso, instrucción a instrucción, cada vez que lo encendemos. No hay misterio: para el Spectrum todos los chips de memoria de su interior (porque hay varios chips, no sólo 2) son como si fuera un gran baúl de 64KB continuados, algo que se consigue mediante cableado de los diferentes chips a las patillas correctas del microprocesador (como hemos visto en el apartado dedicado a la Memoria). A grandes rasgos, las patillas de datos y de direcciones del microprocesador están conectadas a los diferentes chips de memoria de forma que cuando el micro lee datos de la memoria, lo ve todo como si fuera un sólo chip de memoria de 64KB. Esto se consigue con un sencillo proceso de diseño (al hacer el esquema del ordenador antes de fabricarlo) conocido como "mapeado de memoria".

En el mapa de memoria del Spectrum, los primeros 16KB son la ROM (que está en un chip aparte, pero que como acabamos de ver es algo que el Spectrum no distingue, ya que la visualiza como una sección de memoria continua desde la posición 0 hasta la 16383 de su "baúl total" de 64KB) y luego viene la RAM, a partir de la posición 16384. Ahí es donde se almacenan los programas, los gráficos de la pantalla (en un trozo determinado de esa memoria), etc. En esta RAM es donde el intérprete de BASIC introduce los programas para su ejecución.

Estos programas pueden entrar desde los diferentes dispositivos de entrada/salida (gestionados por el Z80) como el teclado, la cinta o disco, etc.

Cabe hacer una mención especial (como ya hemos visto) a que una parte de la memoria RAM (desde el byte 16384 hasta el 23296) está conectada con la ULA, el chip "gráfico" del Spectrum, y encargado de convertir el contenido de esta "videoram" o VRAM a señales de vídeo para la televisión. Cuando los juegos dibujan gráficos, sprites o cualquier otra cosa en pantalla, en realidad están escribiendo bytes en estas posiciones de memoria, que la ULA muestra en la TV en el siguiente refresco de la pantalla.

Así pues, nuestro Z80 en el momento del arranque lo que hace es comenzar a ejecutar uno a uno los opcodes que hay a partir de la dirección 0000h de la memoria, que se corresponde con la ROM. ¿Y qué es la ROM? No es más que un programa realizado por la gente que creó el Spectrum. Ese programa es, entre otras cosas, el intérprete BASIC. Los señores de Sinclair programaron un intérprete BASIC en lenguaje ensamblador de Z80, lo ensamblaron con un ensamblador de Z80 y grabaron el código binario resultante ensamblado en un CHIP ROM de 16KB. Por eso al encender nuestro Spectrum aparece el intérprete de BASIC; el Sistema Operativo de nuestro ZX. Nada nos impediría realizar nuestro propio "sistema operativo" para Spectrum creando una ROM nueva (mirando siempre la compatibilidad con la ROM vieja, de forma que contenga las mismas rutinas de ROM y variables en memoria que utilizan muchos programas) y reemplazando el chip ROM del Spectrum por nuestro propio chip de ROM.

OPCODES Y CÓDIGO MÁQUINA

Nuestro microprocesador Z80 no entiende los comandos en ensamblador que hemos estado viendo en estos 2 primeros capítulos del curso de código máquina; el Z80 sólo entiende números binarios, números de 8 bits de 0 a 255 (o de 00h a FFh en hexadecimal).

De entre los registros del microprocesador hay uno

llamado PC (Program Counter o Contador de

Programa), que es el "puntero" que apunta a la instrucción actual que se está ejecutando. Cuando ejecutamos un programa, lo que hacemos es meterlo en memoria (por ejemplo, como cuando en la primera entrega del curso POKEábamos nuestra rutina a partir de la dirección 40.000) y después saltar al inicio del mismo.

Supongamos por ejemplo que pokeamos el siguiente programa en la dirección 40000:

```
LD A, 0
INC A
LD B, $FFh
INC B
LD DE, $AABB
RET
```

Si ensamblamos este programa obtendremos los siguientes números (técnicamente llamados "código máquina"):

```
3e 00 3c 06 ff 04 11 bb aa c9
```

Al pokear en memoria estos valores, dejaremos la memoria así:

Dirección	Valor
40000	3e
40001	00
40002	3c
40003	06
40004	ff
40005	04
40006	11
40007	bb
40008	aa
40009	c9

Para nosotros estos números no quieren decir nada, pero para el Spectrum tienen un total significado. Concretamente:

Dirección	Valor	Significado
40000	3e	LD A,
40001	00	00h
40002	3c	INC A
40003	06	LD B,
40004	ff	FFh
40005	04	INC B
40006	11	LD DE,
40007	bb	BBh
40008	aa	AAh
40009	c9	RET

A la hora de ejecutar el programa, nuestro RANDOMIZE USR 40000 lo que hace en realidad es cambiar el valor del registro "PC" del microprocesador. Hace PC igual a 40000. Así, el "bucle" del programa que hemos visto arriba en pseudocódigo lo que hace es:

- Leer el byte contenido en la dirección de memoria "PC" (40000).
- Incrementar PC (PC=PC+1).
- El byte es "3eh", con lo cual el Spectrum sabe que tiene que meter en A un valor numérico.
- El valor extra para "LD A," está a continuación en memoria, así que se lee la memoria de nuevo:
    - operando = [PC] = 00h
    - Incrementar PC (PC=PC+1)
- Ya se tiene el "código de instrucción completo", así que se ejecuta: "LD A, 00". (se ejecuta el microcódigo correspondiente dentro de la CPU).

Esto que hemos visto es el proceso de "Lectura de Instrucción (fetch)", "decodificación (decode)", y "ejecución (execute)". Pero recordemos que este proceso se ejecuta una y otra vez, sin parar, de modo que el procesador sigue con la siguiente instrucción (INC A):

- Leer el byte contenido en la dirección de memoria "PC" (40002).
- Incrementar PC (PC=PC+1).
- El byte es "3ch", con lo cual el Spectrum sabe que tiene que incrementar A.
- No hacen falta operandos extra, INC A no requiere nada más.
- Ya se tiene el "código de instrucción completo", así que se ejecuta: "INC A".

Y este ciclo se vuelve a repetir, una y otra vez, hasta que llegamos al RET:

- Leer el byte contenido en la dirección de memoria "PC" (40009).
- Incrementar PC (PC=PC+1).
- El byte es "c9h", con lo cual el Spectrum sabe que tiene que hacer un RET.
- No hacen falta operandos extra, RET no requiere nada más.
- Ya se tiene el "código de instrucción completo", así que se ejecuta: "RET".

Un par de detalles a tener en cuenta:

- Como veis, el microprocesador no entiende el lenguaje ensamblador, sólo la traducción de este a Lenguaje Máquina (los números u opcodes que estamos viendo).
- La primera parte leída de la instrucción es el OPCODE (código de operación), y es lo que permite al Spectrum, mediante una "tabla interna", saber qué tarea exacta tiene que realizar. Si la instrucción necesita datos extra para leer de memoria, se almacenan tras el opcode, y se conocen como "operandos". Así, "LD A,

00" se corresponde con la instrucción "3E 00", donde "3E" es el código de operación (opcode) y "00" es el operando.

- Cuando un operando es de 16 bits (2 bytes), primero encontramos el byte bajo y luego el byte alto. Así, nuestro "LD DE, $AABB" no se codifica como "11 AA BB" sino como "11 BB AA". El opcode para "LD DE" es "11", y "BB AA" los operandos (en este caso, un valor numérico directo). Esta forma de almacenamiento se denomina técnicamente "little endian".
- Para el Spectrum, no hay diferencia entre instrucciones y datos. Un "3Ch" puede ser un "INC A" o un valor númerico "3Ch". ¿Cómo distingue el Spectrum uno de otro? Sencillo: todo depende de si se encuentra al principio de un ciclo de decodificación o no. Es decir, si cuando vamos a empezar a leer una instrucción leemos un "3Ch", es un INC A. Pero si lo leemos en el proceso de lectura de un operando, su significado cambia. Pensad en por ejemplo en "LD A, 3Ch", que se codificaría como "3E 3C", pero no ejecutaría un INC A porque la lectura del "3Ch" se realiza como operando para el "LD A,".
- Al no existir diferencia entre instrucciones y datos, si cambiamos PC de forma que apunte a una zona de la memoria donde hay datos y no código, el Z80 no se enterará de ello y tratará de ejecutar los números que va leyendo como si fuera código (con resultados impredecibles, seguramente con el cuelgue del Spectrum o un reset).
- Por último, existen una serie de opcodes compuestos (dejando de lado los operandos) que ocupan más de 1 byte. Esos opcodes suelen comenzar por CB, ED o FD, de forma que, por ejemplo el opcode "CB 04" se corresponde con la operación "RLC L". Si sólo pudiéramos utilizar un byte para representar el opcode, sólo tendríamos disponibles 256 posibles instrucciones en el procesador. Para poder disponer de más instrucciones se utilizan códigos de instrucción de más de un byte. Así, cuando nuestro procesador encuentra un CB, ED o FD sabe que el próximo código que lea después tendrá un significado diferente al que tendría sin el CB, ED o FD delante. Es por eso que "04h" significa "INC B", y "CBh O4h" significa "RLC L" (otra instrucción diferente).

ENSAMBLADO MANUAL

Al igual que el Spectrum consulta una tabla interna para saber a qué instrucción corresponde cada número (cada opcode) nosotros podemos consultar una tabla para ensamblar manualmente nuestros programas en ensamblador y obtener los valores de código máquina. Si no tenemos a mano un programa ensamblador que lo haga por nosotros (que, al fin y al cabo, no es más que un traductor con una tabla similar), podemos utilizar tablas para traducir el programa manualmente. Cabe decir que es una labor repetitiva y larga, y se recomienda encarecidamente la utilización de un programa ensamblador para ello.

Cuando lleguemos a la parte de definición del lenguaje veremos que ensamblando manualmente resulta bastante costoso calcular las direcciones de los saltos relativos y absolutos, cosa que el programa ensamblador hace con bastante facilidad.

Aún así, quien quiera intentar ensamblar manualmente podrá hacerlo incluso con la tabla de "Juego de caracteres" que tiene disponible en el manual del +2A/+3, capítulo 8, sección 28 (además de no ser la única tabla de ensamblado manual que existe, ya que hay varias disponibles en Internet).

Tabla de opcodes

TIEMPOS DE EJECUCIÓN

Cada instrucción necesita un tiempo diferente para ejecutarse. No es lo mismo un simple "INC A", que requiere leer un único byte como opcode, no requiere parámetros, y sólo realiza un incremento en un registro, que un complejo "LD A, (1234h)", que requiere leer el opcode, a continuación leer 2 bytes para el operando "1234h", después acceder a la memoria y extraer el dato contenido en (1234h) para, finalmente, depositarlo en A.

Los tiempos de ejecución de cada instrucción son, pues, diferentes, y para conocerlos tendremos que consultar cualquier tabla de tiempos (t-states o t-estados). Podéis acceder a alguna de estas tablas en los enlaces que veréis al final de este artículo.

EL SOFTWARE DE SPECTRUM

A estas alturas ya debemos tener claro cómo funciona el Spectrum, con su microprocesador Z80 continuamente ejecutando el código apuntado por "PC", incrementando este y de nuevo repitiendo el ciclo.

Cuando encendemos nuestro Spectrum, PC vale 0000h y se ejecuta la ROM que, como ya hemos comentado, no es más que un programa hecho por los ingenieros que desarrollaron el Spectrum. Dicho programa está disponible en formato código fuente para consultas ya que usuarios de Spectrum realizaron un desensamblado (a partir de los opcodes, obtener el código fuente original) y comentaron todas las rutinas, variables del sistema y procedimientos que se ejecutan en

nuestro Spectrum nada más arrancarlo. El libro "The Complete Spectrum ROM Disassembly" (El desensamblado Completo de la ROM del Spectrum) contiene este desensamblado, y podemos obtenerlo en Internet, por si tenemos curiosidad en conocer las interioridades de la ROM del Spectrum y cómo está programado el intérprete BASIC y las diferentes funciones de la ROM del mismo (con el objetivo de poder "usarlas" en nuestros programas en ensamblador).

Pero aparte de la ROM del Spectrum, ¿cómo llega a la memoria de nuestro ordenador (o emulador) los programas que ejecutamos?. Veamos las diferentes maneras:

1. Desde cinta: Nuestro LOAD "" provoca en BASIC la llamada a una rutina de la ROM que carga desde cinta el código y los datos de los programas. Lo único que se hace es leer de la cinta los opcodes y sus operandos, así como cualquier otro dato (gráficos, sonidos) del programa, e introducirlos en memoria en una zona a la que luego saltaremos (cambiaremos PC a ella). Cuando grabamos a cinta, lo que hacemos es leer el contenido de un trozo de memoria y escribirlo en cinta (escribir los valores numéricos de los opcodes, operandos y datos).
2. Desde disco: exactamente igual que en el caso de la cinta, pero el medio de almacenamiento es un disco de 3" o de 3.5".
3. Ficheros TAP y TZX: son ficheros de ordenador que almacenan los datos exactamente igual que si fuera una cinta real: almacenan opcodes, datos y operandos, que luego serán cargados en memoria.
4. Ficheros .SP, .SNA y .Z80 (en general, cualquier fichero de snapshot). No son más que volcados de la memoria. Por ejemplo, un fichero .SP o .SNA contiene el contenido de las 49152 celdillas de memoria desde 16384 hasta 65536. Para cargar ese .SNA en un emulador, lo que realiza el emulador es un simple "POKEado" del contenido del fichero en las celdillas de memoria. Así, un fichero snapshot no es más que una "copia" de la memoria (de su contenido) que volcamos a fichero.

EN RESUMEN

Hemos visto cómo funciona internamente nuestro ordenador Spectrum y el microprocesador Z80. A partir de la próxima entrega comenzaremos ya con la sintaxis del lenguaje ensamblador y una descripción de las diferentes instrucciones disponibles. No obstante, creemos que los conceptos introducidos ya en estas 2 primeras entregas del curso deben de haber llevado ya al lector a un punto en el cual podrá realizar sus primeras pruebas en ensamblador mediante la documentación a la cual nos referimos en los enlaces. Basta con consultar el juego de instrucciones del Spectrum en la página oficial del Z80 o de Zilog para poder realizar ya nuestros primeros programas en ensamblador para nuestro querido Sinclair ZX Spectrum.

FICHEROS

Programa en BASIC que lee el teclado mediante acceso a puertos

Fichero tap del ejemplo teclado.bas

Programa en BASIC que dibuja en pantalla mediante POKE

Fichero tap del ejemplo pantalla.bas

Listado de opcodes para ensamblado manual

LINKS

Puertos E/S

Usando CM

La Memoria

Variables del sistema

Set de caracteres

Web del Z80

Z80 Reference de WOS

Z80 Reference de TI86

Tablas de ensamblado y t-estados (pulsar en Z80.txt, Z80_reference.txt, Z80time.txt)

Complete Spectrum ROM disassembly project

SROMERO

AVENTURAS Y DESVENTURAS DEL VIEJO ARCHIVERO

"Noche de paz y tranquilidad en los Cárpatos. Sólo el reconfortante aullar de las jaurías de lobos a la luna llena que se asoma tímida entre los densos nubarrones. En el más alto pico se levanta, entre profundos cortados y rodeado de niebla, un vetusto, pero señorial castillo". Así comenzaba el primer artículo de El Viejo Archivero, carismática y añorada sección de la revista MicroHobby. Sólo tres meses antes, su autor Andrés Samudio había abierto su otra ventana en MH. Era otoño de 1987. Y el joven mundo de la aventura en España, aún tambaleante, se desperezaba risueño junto al venerable anciano.

Que la aventura conversacional siempre ha sido un género minoritario en España es un hecho comúnmente asumido. Nada comparable, allá en los 80, al movimiento existente en Gran Bretaña, donde grandes compañías especializadas se dedicaban al desarrollo profesional de aventuras y parsers con los que deleitaban a su vasta legión de seguidores. Aquí, los pocos afortunados que habían descubierto la riqueza del mundo aventurero devoraban con avidez, diccionario en ristre, las joyas británicas que lograban cruzar nuestras fronteras. La escasez era la norma; y sólo una compañía de la envergadura de Dinamic se permitía hacer ciertas incursiones en el campo.

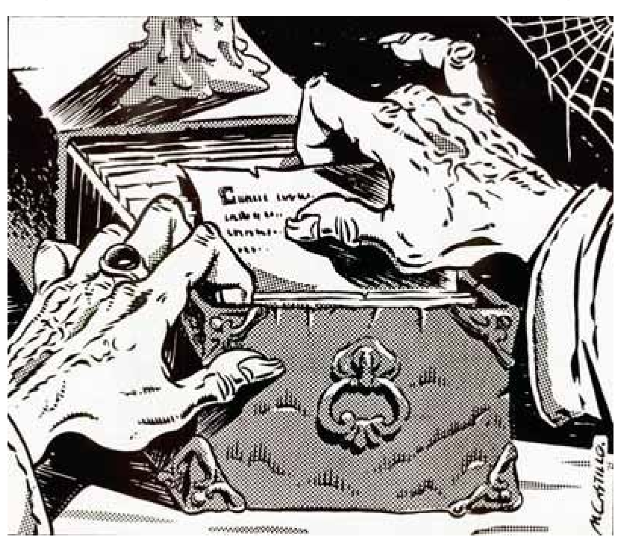


Ilustración que acompañaba al primer artículo de "El Viejo Archivero" (MH 158)

Viajemos, pues, en el tiempo hasta comienzos del año 1987. Por aquel entonces, el usuario español de Spectrum sólo había podido disfrutar de cuatro aventuras en castellano. Abrió la cuenta el pionero "Yength", publicado por Dinamic en sus albores (1984) y programado por Nacho Ruiz. Tras pasar sin pena ni gloria, fue MicroHobby la que puso el siguiente granito, lanzando en uno de sus cassettes "Alicia en el país de las maravillas", obra de Luis E. Juan. Paralelamente, los aventureros podían degustar uno de los pocos títulos traducidos al castellano: el excelente "Gremlins: The Adventure", de Adventuresoft, que para muchos se convertiría en su primer contacto con el género. Tan desolador panorama se complementaba con "Cobra's Arc", programa lanzado en 1986 por Dinamic y que, sin ser una conversacional al uso en su sentido más estricto, resultó un aceptable éxito de ventas.

Sin embargo, algo iba a cambiar en 1987 para el desamparado aventurero español. A raíz, principalmente, de dos sucesos. En primer lugar, el lanzamiento de las aventuras Arquímedes XXI y, sobre todo, Don Quijote (fenómeno en el que se profundiza en otra sección de este mismo número de MagazineZX). El segundo suceso, mucho menos ruidoso pero no por ello de menor importancia, fue la aparición de Andrés Samudio como activo colaborador en las páginas de MicroHobby, tan sólo mes y medio después de que el ingenioso hidalgo pasase a engordar el catálogo de Dinamic.

Samudio (quien más tarde engendraría Aventuras AD) abría en el número 146 una nueva sección de la revista, titulada "El Mundo de la Aventura", en la que desgranaría paulatinamente los más preciados secretos sobre la aventura conversacional y su proceso creativo, con gran mimo y un espíritu eminentemente didáctico. Además, en ese mismo número realizaba un llamamiento a sus avezados jugones: "¡Consúltanos tu problema! y envíanos tus cartas, indicando en el sobre: <<ARCHIVOS DEL AVENTURERO>>".

Doce números más tarde, Samudio contestaba a las primeras cartas llegadas a la redacción bajo una nueva sección, que tituló "El Viejo Archivero". En dicho número, el 158, se encarnaba por vez primera en el homónimo personaje, un decrépito anciano habitante de un castillo transilvánico, y contestaba a las preguntas de los lectores sobre el juego Spiderman de la serie Questprobe, a la cual dedicaría las siguientes ediciones de "El Viejo Archivero". En un tiempo sin Internet, los aventureros españoles dejaban de ser seres aislados, y pasaban a compartir sus dudas e inquietudes a través del espacio que Samudio les había abierto.

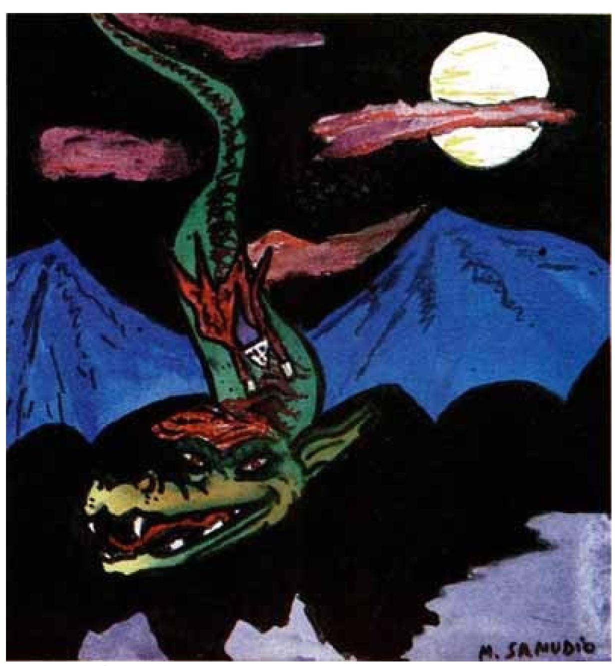
EL VIEJO ARCHIVERO

La rica imaginería ideada por Samudio rápidamente calaría entre los lectores de la revista, permitiendo la aproximación de nuevos jugones al mundo de la aventura, atraídos al encanto de las imaginativas historias y la fina ironía que destilaba en sus escritos. Curiosamente, muchos de los que recordamos el Viejo Archivero lo hacemos no tanto por las respuestas que Samudio cuidadosamente daba a las cuestiones de los lectores, sino por los personajes que introdujo para hilvanar la sección, y que la dotaban de un encanto particular.



El abominable Viejo Archivero, en primerísimo plano (MH 192)

El primero en hacer su aparición, en el mismo número 158, fue el dragón Smaug, inmenso ser volador de incendiario aliento y fétidas ventosidades que acompañaría fielmente al Viejo durante su periplo por MicroHobby. El dragón era una suerte de homenaje del autor a "El Hobbit", juego del que era un apasionado entusiasta, así como del mundo de Tolkien, como confesó en múltiples ocasiones desde sus propias páginas. Inclusive, se permitiría la licencia de explicar en el número 208 cómo el Archivero rescató al dragón de su muerte a manos del mismísimo Frodo Bolsón.



El dragón Smaug huye con el Archivero a lomos. Ilustración de Mónica Samudio (MH 208)

Otros personajes aparecerían más adelante. Entre los más recordados se encuentra sin duda el infecto y saltarín Yiepp; en palabras del propio Samudio, perdón, del Archivero, una "especie de injerto abominable entre una saltamontes que ronda permanentemente ebria por los rincones de mi castillo y un libidinoso gnomo". El engendro Yiepp se asomó por primera vez en MicroHobby 189, y no parece descabellado sospechar que hacía referencia a algún integrante parlanchín del equipo de Aventuras AD: Samudio afirmó en más de una ocasión que se trataba de un "verdadero experto en aventuras españolas", con las que a menudo echaba una mano en la sección.

Mejor parada (pero no demasiado) salía la fermosa doncella Hebilla de Calatayud, que se daba a conocer en MicroHobby 192 y que firmaría con el propio Andrés Samudio los últimos números de "El Viejo Archivero". Hebilla tenía la insana costumbre de comerse a los intrépidos aventureros y otros pretendientes, razón por la cual casi siempre estaba royendo algún meñique. Aunque nunca se explicitó en las páginas de MicroHobby, parece bastante obvio que Hebilla era, en realidad, Eva Samitier, la secretaria de Aventuras AD, que pacientemente contestaba a las cartas de los lectores ante el ingente trabajo del director Samudio.

En MH 203 aparecerá un último personaje, bajo el descriptivo epígrafe: "Juanmilla se une a la

Pandilla". Sin datos fidedignos que lo confirmen, el abajo firmante sospecha que Juanmilla era, en realidad, Juan Manuel Medina, programador de los últimos títulos de Aventuras AD. Para finalizar la tourneé familiar, apúntese que las ilustraciones de las dos secciones de Samudio eran, a menudo, realizadas por sus dos hijas: las gemelas Mónica y Guisela Samudio. Concretamente, la primera en publicar uno de sus dibujos es Guisela, en MicroHobby 169, para la sección "El Mundo de la Aventura", a la que frecuentemente solía dedicarse (quedando así para Mónica la parte gráfica de "El Viejo Archivero").



Hebilla, Yiepp y el Viejo, según visión de Mónica Samudio (MH 195)

Haciendo gala del mismo humor con que a menudo trataba a sus compañeros, Samudio se abstuvo siempre de ser autoindulgente. La nómina de calificativos que aplicó a su alter ego, el Archivero, era simplemente apabullante: "el carcomido momio", "El Agonizante", "el cuasi-momificado", "baboso vejestorio", "esperpento de los Cárpatos", "viejo adefesio", "desdentado tatarabuelo", "destartalada momia", "decrépito anciano", "el peripatético", "el Obsoleto", "multiesclerótico", "agónico", "medio infartado", "pitecantropus totémico", "la más esmirriada, encorvada, temblorosa y carcamálica figura", "cascarrábico esperpento", "apolillado", "el Vetusto" o "el Carcamal".

Otra de las bromas que solía divertir a los seguidores de MicroHobby la llevaba a cabo a costa de su propio nombre. En sus primeros artículos, siempre firmaba como Andrés R. Samudio. Hasta que a algún curioso lector se le ocurrió preguntarle a qué correspondía la inicial 'R'. El ingenioso Samudio pidió a sus lectores que contuviesen la risa, pues su verdadero nombre era "Andrés Requiescantinpace Samudio". Ante la algarabía general, Samudio fue dando nuevo significado a la R en cada uno de sus artículos de "El Mundo de la Aventura", firmando como: Rififi, Rufo, Repelente, Rojillo, Rambo, Rebombón, Retrete, Radioactivo, de Reojo, Reagan, Rumiante, Rollero, Recochineo, Revoltillo ó Refrescante. En MicroHobby 211, con motivo del fallo del Concurso de Aventuras de MicroHobby, Samudio cambia su firma por un cuasi-protocolario "Andrés Respetuosamente Samudio Monro". En MicroHobby 212, publica la entrevista a los ganadores del susodicho concurso, y al fin da a conocer su verdadero nombre: Andrés Roberto Samudio Monro. Sería el último artículo de "El Mundo de la Aventura", quedando "El Viejo Archivero" como último reducto hasta la desaparición de la revista en enero de 1992.

SAMUDIO Y LA AVENTURA 'HOMEGROWN'

El Concurso de Aventuras de MicroHobby (celebrado a lo largo de 1990, y patrocinado por Aventuras AD) fue en sí un hito dentro del panorama aventurero español. Aunque Samudio se ha confesado públicamente insatisfecho con el lento proceso de selección y su conclusión (véase la entrevista para Computer Emuzone), parece claro que fue un fuerte punto de apoyo para el pujante mundo de la aventura 'homegrown' (artesanal, o amateur), que venía fraguándose desde la creación del "Club de Aventuras AD" (CAAD).

Por aquel entonces (90-91), ya existían un buen número de fanzines y clubs a través de los cuales los aventureros podían ponerse en contacto y compartir sus obras. Entre los más destacables, se podrían citar el valenciano CAAD, el gallego "Z For Zero", o el zaragozano "A Través del Espejo". El fanzine más longevo, aunque ya desaparecido, fue el del CAAD, persistiendo hoy en día sustitutos en la red como SPAC. El club del CAAD (que no el fanzine), sigue sin embargo existiendo, y su página web es visita cuasi obligada para los irreductibles aficionados a la aventura.



Primer dibujo de Guisela Samudio en MicroHobby, en la sección "El Mundo de la Aventura" (MH 169)

Viene esto al hilo de nuestra historia porque la creación del CAAD se anunció por primera vez en MicroHobby 171, en la sección "El Mundo de La Aventura", en el ya remoto junio de 1988. Anteriormente comentábamos que la irrupción de Samudio en MicroHobby supuso la apertura de una ventana al mundillo aventurero: con este anuncio, se dio paso a la libre comunicación entre sus seguidores, y como consecuencia al crecimiento de la aventura homegrown en España (gracias también, cómo no, a la aparición de parsers en castellano). De algún modo, se abría camino a un fenómeno que ya ebullía en las islas británicas.

Es de justicia mencionar aquí el importantísimo papel jugado por Juan José Muñoz Falcó, compañero de Andrés Samudio en Aventuras AD, y creador del CAAD. La oportuna iniciativa cristalizó en el primer fanzine del CAAD, dirigido y editado por el propio Juanjo Muñoz, y anunciado por Samudio en las páginas de MicroHobby, concretamente en el número 187.

Mientras tanto, muchos insignes aventureros iban apareciendo entre las páginas de MicroHobby, a menudo realizando consultas en "El Viejo Archivero". Muchos de estos audaces, capaces de someterse a las más complejas aventuras en inglés, se convertirían en nombres familiares por sus propias obras, perdurando algunos en activo hasta nuestros días. Como dicen que "para muestra, un botón", remitiremos al lector a las preguntas de SCP Hackers (en MicroHobby 166, sobre "The Fantastic Four"), Javier San José (en MicroHobby 186, preguntando sobre "Bored of The Rings"), o Josep Coletas Caubet (en MicroHobby 192, sobre "Abracadabra").

El tercer gran anuncio que realizaría Andrés Samudio se produjo ya en febrero de 1990 (MicroHobby 196), relativo a la creación de la bolsa de aventuras del CAAD. Paralelamente, se había lanzado el concurso de aventuras de MicroHobby y Aventuras AD (quedaban dos meses para el cierre del plazo de entrega de trabajos). Y Cozumel, para muchos la mejor aventura jamás publicada por Aventuras AD, estaba a punto de ver la luz.

CONCLUSIÓN

Hacía dos años y medio, en aquel febrero de 1990, desde que Andrés Samudio se había asomado a las páginas de MicroHobby. Durante ese tiempo, el mundo de la aventura había madurado, y se había acercado a un perfil muy distinto de usuario. Esto queda reflejado en el sustancial aumento de aventuras comerciales publicadas desde entonces, de muy distintas compañías. Para Spectrum, no menos de doce nuevas aventuras ("Megacorp", "La Guerra de las Vajillas", "Carvalho", "Post Mortem", "La Corona", "Abracadabra", "La Aventura Original", "Zipi y Zape", "Ke Rulen Los Petas", "Corrupt", "Jabato" y "Legend") a la que podríamos añadir "Supervivencia (El Firfurcio)", título menor de Aventuras AD publicado por la propia MicroHobby.

Quizá inconscientemente, Samudio estaba dando paso en aquel momento a una nueva forma de entender la aventura, fuera de los circuitos comerciales, ante el advenimiento de una nueva época en la que otro tipo de videojuegos llevarían, aún más, la voz cantante. Las bolsas de los clubs, entre ellas la del CAAD, se fueron llenando de excelentes obras, algunas de las cuales aún están por descubrir para muchos de nosotros. Aventuras AD permanecería como la única defensora de la aventura bajo el ala de Dinamic, lanzando la trilogía del Yucatán y "La Aventura Espacial", antes de que la propia Dinamic la arrastrara en su caída, en el año 1992.

Hoy en día, la aventura sobrevive en nuestro país gracias al encomiable y maravilloso esfuerzo de tantos aficionados que aman el género y no lo han dejado morir. Parsers, librerías y aventuras siguen apareciendo gracias y para el deleite de sus acólitos. Mientras tanto, Andrés Roberto Samudio Monro volvió a su vida de pediatra, alejado del mundillo, y sus gemelas Mónica y Guisela son hoy en día periodista y economista (o viceversa, tanto da). Sin embargo, Samudio puede albergar en su fuero interno el noble orgullo de haber insuflado vida a los sueños de muchos de nosotros.

Y ahora, dejemos descansar al Viejo hasta la próxima luna.

LINKS

MicroHobby Nº 171 [Junio 1988]

Aventuras AD en Macedonia Magazine

Entrevista a Juanjo Muñoz en MM

Computeremuzone [Julio 2001]

Página del CAAD

FAQ del CAAD

Proyecto BASE

JUAN PABLO LÓPEZ-GRAO



DESILUSIONES E ILUSIONES DE UN AÑO PASADO

Es ésta una época del año propicia para hacer resúmenes de lo acontecido en los doce meses anteriores. Los diferentes medios de comunicación nos ofrecen docenas de artículos mostrándonos los higadillos de los famosos, los mejores goles del año, las frases más desafortunadas de los políticos a los que luego debemos votar, o las imágenes más macabras de las desgracias acaecidas, entre otras colecciones de hechos y sucesos que nos pueden importar mucho, o no.

Para Magazine ZX no iba a ser una excepción. En el entorno del Spectrum han sucedido bastantes cosas dignas de mención, y otras muchas indignas de ello. No pretendo hacer un repaso exhaustivo de todo, seguro que me puedo dejar algo en el tintero, pero sí quiero sobrevolar y dar mi opinión sobre algunos de ellos.

Como hoy me he levantado con la vena masoquista ligeramente marcada en la frente, voy a empezar por hacer acto de contrición sobre la publicación que estáis leyendo. Y de paso aprovecho para curarme en salud y que no me puedan reprochar, al término de la lectura si es que alguien llega al final, que veo la paja en el ojo ajeno y no la viga en el propio.

Aquél 2005 terminado hace unos días ha sido un año difícil para esta publicación. De la periodicidad mensual con la que comenzamos en el verano del 2003, pasamos a publicar cada dos meses ante la imposibilidad de mantener el ritmo mensual y además tener una vida al margen del Spectrum. Pero ese año con rima obscena supuso un punto de inflexión. Lo comenzamos con un mes de retraso sobre el número anterior y lo continuamos con una acumulación de siete meses más para la segunda y última edición. Edición que estuvo muy cerca de suponer el fin de esta pequeña publicación ‘on line’. Se pueden alegar muchas excusas. Trabajo, cansancio mental, falta de inspiración, etc. Pero la realidad, por lo menos en el caso del que esto escribe, es que es la pereza y la dejadez la causante de muchas impuntualidades y de la mala calidad en algún artículo, texto que te tienes que obligar a repetir ya que no serviría ni para publicarlo en un “inframedio gratuito” . Y es que hay momentos en que no te apetece dejar de hacer una actividad determinada en tus ratos de ocio para ponerte a jugar al juego que toque analizar. Que seguro te encanta pero para el que no es el momento apropiado. Estados de ánimo. El lado positivo, ironizando por supuesto, es que este año tenemos que imprimir menos números para regalar al final de la MadriSX & retro 2006.

El año se puede decir que comenzó con la MadriSX & retro 2005. Desde mi punto de vista todo un éxito en lo tocante al mundo Sinclair al menos y con la asistencia de algunos ilustres programadores y grafistas de la época dorada: Marcos Jourón, Fernando Sáenz, Carlos García Cordero o Alfonso Fernández Borro. MZX junto con Speccy.org montó un stand con dos mesas repletas de equipos, periféricos y software. El stand estuvo bastante frecuentado por visitantes ocasionales y por viejos conocidos del mundillo, y tuvimos el honor de contar con la colaboración inesperada de Julio Medina y sus ‘gadgets’ para el Spectrum.

Agradable también fue el tremendo éxito del stand de SPA2, con Juan Pablo a la cabeza, en su tarea de preservar software. La afluencia de gente que llevo sus cintas fue amplia, no se pudo hacer toda la tarea y muchas cintas se prestaron para seguir con el trabajo posteriormente. Mención especial en este apartado merece la recuperación de un juego inédito anunciado en su época por Dinamic y ofrecido desinteresadamente por sus autores en esta reunión. Este tema lo trataremos posteriormente.

El stand que presentaron Javier Guerra ‘Badaman’ y Juanjo, dedicado íntegramente al QL, “hermano mayor” de la familia Sinclair, era una delicia, especialmente para los que desconocíamos muchos detalles de una máquina más conocida en superficie que por lo que es capaz de hacer.

No olvidemos de mencionar a Ceinsur, con Miguel A. Montejo “Radastan” al frente. Con un stand centrado en la venta de material retro y en una “sala de juegos” en la que habían montado varios equipos, mayormente de la marca Amstrad, para que la gente se echara unas partidillas. Hay que hacer mención a la cesión por parte de Miguel de un amplio lote de cintas para su preservación en SPA2. Tampoco podemos dejar de lado a Ssergio Vaquer “Beyker” que llevó una edición en cinta de sus juegos con un acabado profesional listos para la venta.

Otro tema aparte merecen las nuevas producciones de software, lúdico al 101%, realizadas el año pasado en nuestro país. Este campo lo podemos diferenciar en dos frentes. Por un lado los juegos que realmente son nuevos, es decir, programados en la actualidad y por otro las "viejas producciones" que ven la luz 20 años después.

En el primer caso la producción ha sido amplia. Comenzando con los juegos presentados a los diferentes concursos que se convocan desde Bytemaniacos. Juegos limitados por las propias reglas impuestas, algo lógico, lo que no quita mérito ni a sus autores ni a la convocatoria de los mismos concursos, con los que se consigue despertar el "gusanillo" a los aficionados a la programación y pasar un buen ratillo tanto jugando como escribiendo código.

Otros dos títulos se presentaron desde Compiler Soft: ZX Columns y Another brick on the wall 2. El primero una versión del clásico de Sega y el segundo un machacaladrillos clásico. Con poca originalidad en su concepto.

Desde Computer Emuzone las realizaciones fueron varias destacando Moggy y Columns. Ya que, aunque sacaron más títulos, estos no dejaban de ser versiones modificadas de los mismos juegos presentados a los diferentes concursos que comentamos antes. Mención especial merece el Columns, un juego que por su calidad y presentación bien podría haber sido un lanzamiento comercial de los años ochenta. Sin querer restar ningún mérito a este grupo de programadores, me ha causado una muy mala impresión la reacción a los diferentes comentarios realizados por los usuarios en el momento del lanzamiento de los juegos, en especial a los que avisaban de fallos o "bugs" en los programas, por parte de los componentes de CEZ, contestando con muy malas formas, tanto en medios como las NEWS o en su propio foro, en una actitud prepotente que recuerda bastante al gigante Microsoft con sus famosas "es una "feature", no es un bug" y que a mi en particular me hacen perder el interés, salvo el que pueda tener para dar información en este magazine, en sus producciones. No es un hecho aislado, con el lanzamiento hace unos días del espectacular remake de Sir Fred, realizado por Celemin, se han repetido los hechos por un "me estás robando unos clicks" contra la WEB Remakes Zone y unas cláusulas restrictivas en la licencia de distribución que no tienen sentido en un grupo de personas que se supone lo hacen por amor al arte, aunque están en su derecho de hacer lo que les cuadre con sus programas, no cabe duda.

Referente a la programación, y como implicado en ella, tengo la impresión de que el nivel actual en nuestro país es bastante más bajo que al principio de los 80. Los programas que se realizan, a excepción del Colums de CEZ, no serían comercializados por ninguna compañía seria en su momento. Y hay cosas que son difíciles de explicar, ya que ahora, con los medios de que disponemos: compiladores cruzados, emuladores, programas de diseño gráfico, todo ello corriendo en PC's potentes, y la información que tenemos disponible de la máquina, lo normal sería crear juegos de buen nivel y los proyectos que parecen más serios, y que llevan anunciados durante años, siguen siendo 'vaporware', lo que parecían en sus inicios.

Siguiendo con los juegos, pero en la categoría "nunca publicados en los 80", la situación ha sido excelente en este año pasado. Por un lado en MadriSX 2005, y gracias a Josetxu Malanda 'Horace' que contactó con sus autores, vio la luz 'Vega Solaris', un juego anunciado por Dinamic y nunca publicado. Sus autores: Fernando Sáenz y Carlos García Cordero, llevaron los 'masters' con el original del juego y bastante documentación sobre el código. En una dura tarea de preservación se consiguió rescatar y gracias a ello podemos disponer de esta pequeña maravilla del software lúdico, y con ello dejamos sin respuesta una pregunta que viene a la cabeza de inmediato al ver el juego: ¿cómo es posible que no se publicara el juego?

Por otro lado se rescataron varios juegos inéditos de los hermanos Vives: El paso, Sigfrido y Frighful. Oscar y Enrique Vives fueron los autores de Mambo o el Amo del Mundo entre otros. Y ya para terminar Slowglass, de Alberto Pérez y Manuel Domínguez, autores en su tiempo de Cyberbig.

Y dejando de atrás la creación de software, nos metemos de lleno en lo que es el movimiento actual de la "escena" del Spectrum.

Públicamente los usuarios de Spectrum en habla española tienden a moverse alrededor del grupo de news es.comp.sistemas.sinclair, aunque prefiero pensar que el estado actual de dicho grupo no es el reflejo de la situación de la escena. Y prefiero no pensarlo porque, aunque cuantitativamente el número de mensajes es abundante, cualitativamente es similar a las creaciones de los guionistas de 'Salsa Rosa'. Y es que ya no recuerdo el tiempo que hace que no se crea un hilo interesante, bien sea sobre temas técnicos, de culturilla histórica sobre la máquina o cualquier tema que pueda despertar un mínimo interés. Y reflejo de ello es que cada vez escriben

más ‘casuals’ que entran para preguntar por un tema específico, normalmente donde se puede descargar tal juego o anuncian a bombo y platillo que van a desarrollar el ordenador de 8 bits definitivo, y alguno de ellos siguen posteando durante unas semanas hasta desaparecer igual que llegaron. Seguro que hablar de asuntos intrascendentes, del color de los ojos de Sabreman o del interfaz del Three Weeks in paradise le interesa a mucha gente, pero a mi me deja indiferente y es lo que hace que cada vez entre con menos frecuencia a pegar un vistazo por esos lugares.

Muchos temas me dejo en el tintero, como pueden ser las actualizaciones de las páginas WEB ´’clásicas’, la creación de otras nuevas que duran poco, el mercadeo de artículos, el sobre precio que se llega a pagar por lotes de cintas en Ebay y un largo etcétera, pero todo esto, lo dejamos para otra ocasión. Ahora a seguir pasándolo bien con nuestros Spectrum’s y a pensar en la próxima reunión en MadriSX & Retro 2006.

Feliz año.

MIGUEL


ZX
magazine